applied sciences

MDPI

Article

Metamaterial Design with Nested-CNN and Prediction Improvement with Imputation

Erkan Kıymık [1,*] and Ergun Erçelebi [2]

[1] Electrical and Electronics Engineering Department, Hasan Kalyoncu University, Gaziantep 27100, Turkey
[2] Electrical and Electronics Engineering Department, Gaziantep University, Gaziantep 27100, Turkey; ercelebi@gantep.edu.tr
* Correspondence: erkan.kiymik@hku.edu.tr

Abstract: Metamaterials, which are not found in nature, are used to increase the performance of antennas with their extraordinary electromagnetic properties. Since metamaterials provide unique advantages, performance improvements have been made with many optimization algorithms. Objective: The article aimed to develop a deep learning model that, unlike traditional optimization algorithms, takes the desired reflection coefficients' parameter as an input and gives the image of the corresponding metamaterial. Method: An amount of 29,722 metamaterial images and reflection coefficients corresponding to the metamaterials were collected. Nested-CNN, designed for this task, consisted of Model-1 and Model-2. Model-1 was designed to generate the shape of metamaterial with a reflection coefficient as the input. Model-2 was designed to detect the reflection coefficient of a given image of metamaterial input. Created by using Model-2 in Model-1's loss function, the nested-CNN was updated by comparing the reflection coefficient of the produced image with the desired reflection coefficient. Secondly, imputation, which is usually the complete missing data before the process of training in machine learning algorithms, was proposed to use in the prediction side to improve the performance of the nested-CNN. The imputation for prediction was used for the non-interested part of the reflection coefficient to decrease the error of the interested region of the reflection coefficient. In the experiment, 27,222 data were used for the KNN-imputer, half of the reflection coefficient was considered as the non-interested region. Additionally, 40 neighbors and 50 neighbors were given the best mean absolute errors (MAE) for specified conditions. Result: The given results are based on test data. For Model-2, the MAE was 0.27, the R2 score was 0.96, and the mean correlation coefficient was 0.93. The R2 score for the nested-CNN was 0.9, the MAE of nested-CNN was 0.42, and the MAE of nested-CNN with 50 neighbors was 0.17.

Keywords: metamaterial; metasurface; frequency selective surfaces; convolutional neural networks; deep learning

Citation: Kıymık, E.; Erçelebi, E. Metamaterial Design with Nested-CNN and Prediction Improvement with Imputation. *Appl. Sci.* **2022**, *12*, 3436. https://doi.org/10.3390/app12073436

Academic Editor: Krzysztof Koszela

Received: 10 March 2022
Accepted: 28 March 2022
Published: 28 March 2022

1. Introduction

Metamaterials(MM) are special materials that are not found in nature and are used in many fields with their unusual properties [1]. Metamaterials have a wide range of uses, including optical filters, medical devices, remote aerospace applications, sensor detection and infrastructure monitoring, smart solar power management, crowd control, high-frequency battlefield communication, and lenses for high-gain antennas, improving ultrasonic sensors, anti-radar devices, and superlenses. The breadth of these usage areas has been the motivation for many studies on developing metamaterials. The studies are generally based on iteration-based studies of optimization algorithms. Optimization algorithms generally proceed iteratively until optimum and satisfactory results are found. The increase in big data and processing power demonstrates that the future of optimization algorithms will be even brighter. In the past few decades, electromagnetic (EM) designers have rapidly and

accurately developed models for rapid progress in technology. Moreover, these models have allowed for more variety and modeling complex problems than previously imagined.

In recent decades, although many global optimization techniques have been developed, the most used technique is the genetic algorithm (GA) for designing metamaterials. (GA) [2]. GAs mimic natural selection and mutation to optimize constrained and unconstrained problems. The GA's basic operation chooses the best options from the previous generation to breed the next generation; it was popularized and widely used by the design of pixelized metallic structures [3], which is the basis of many metamaterial designs [4]. Optical nanoantenna array configurations [5] and shapes [6] have been optimized to maximum field enhancement at the selected location. In [7], the GA has a decreased field enhancement, which is a prohibitive factor for metamaterial usage in high power microwave implementations and thus, has a reflectarray unit cell design based on pixels. In recent applications, coding metasurfaces based on radar cross-section (RCS) reduction and best suitable metamaterials developed for polarization conversion, has been included by the GA in [8–10]. Many algorithms are created by nature [11,12], such as artificial bee colonies (ABC) [13] and bat-inspired algorithms [14]. These algorithms are known as lower branches of swarm intelligence algorithms that try to optimize complex problems by mimicking the collective behavior of decentralized, self-organized systems, natural or artificial [15]. The optimization of an ant colony (ACO) [16] is counted as one of the swarm intelligence algorithms, and uses stigmergy to find the best suitable solution. Ant colony optimization is used for graph-based solutions, such as finding the route of the vehicle, job scheduling, and traveling seller issues [17]. The GA has achieved immense results in electromagnetics and optical materials, but disjointed pixels in solution are a drawback of GAs because these designs are limited to planar configurations. On the other hand, ACO maps the optimal "trace" found by artificial "ants" in the graph topology to an adjacent structure. For that reason, ACO has achieved enormous success in the design of the electromagnetic device, particularly in the meander-line antennas [18]. For the high performance of frequency selective surfaces (FSS), the ACO algorithm and lazy ants combined and improved the performance of FSS by ZHU [19,20]. In the mid-1990s, another branch of swarm intelligence algorithms, particle swarm optimization (PSO), emerged [21], and has been used immensely for the optimization of electromagnetic devices [22–25]. Compared to PSO and GA, PSO offers many more advantages: firstly, PSO uses real values as vectors instead of binary values, and real values were later used in GAs [26]. Secondly, in PSO, members of the population tend to work more independently and cooperatively. This can be considered as individual members work on solutions of a different part of the space and communicate with each other. PSO gives us better performance than GA for designing negative-index metamaterial [27]. Moreover, PSO is used in meta-optic device optimizations. PSO has been used to optimize geometrical parameters and spacing-based Yagi-Uda antennas [28]. PSO is used in beam steering applications by optimizing nanohole array-based metamaterials [29]. While global optimizers, such as GA and PSO have been successfully applied to a wide variety of design problems in RF and optical regimes, they are often sensitive to internal parameters that require problem-wise adjustment. Optimum control parameter tuning in evolutionary algorithms has been studied by many studies [30–32]. Moreover, it is not always clear how best to set these parameters for a particular problem and may require adaptive tuning during optimization to maximize algorithm performance. In this context, the covariance matrix adaptation evolution strategy (CMA-ES) with a self-adaptive nature, emerges as the answer to the adaptive tuning problem [33]. It was found that CMA-ES performed significantly better than GA in terms of convergence speed and solution quality when the wideband polarization converter was applied to the optimization of metastatic surfaces [34].

The deep neural network (DNN) and convolutional neural network (CNN) move beyond the traditional supervised learning algorithms and optimization methods since computational capacity is improving rapidly day by day [35]. DNN is used in classification and regression problems and has achieved great success. CNN is a DNN algorithm and can

take pictures, matrices and signals as input. The purpose of CNN is achieved by extracting the features with the filters, the coefficients of the filters and biases are updated with gradient-based optimizations. In the creation of metamaterials, the shapes were generally optimized by iteration-based experiments, for example, optimization based on reflection coefficients. In such iteration-based experiments, at best, the duration of the simulation program elapses in each iteration while trying the shapes. Instead of finding reflection coefficients from the shape as traditional iteration-based methods do, a model that provided metamaterial pictures corresponding to the desired reflection coefficient was developed by approaching the metamaterial production with DNN in reverse.

In this article, first, metamaterial design was made by the proposed nested-CNN model with the inverse functioning method. The nested-CNN model consisted of two models: Model-1 was trained to detect the image of MM by using a desired reflection coefficient as the input. Model-2 was trained to detect the reflection coefficient of a given image of MM as the input. Trained Model-2 was used in Model-1's loss function in the training process of Model-1, which compared the desired reflection coefficient and reflection coefficient of the produced image of MM by Model-1. In nested-CNN, the output of Model-1 was the image of the MM; the output of Model-1 was used as the input to Model-2; Model-2's output was the reflection coefficient of the produced image of MM by Model-1; and the input of Model-1, which was the desired reflection coefficient and output of Model-2's obtained reflection coefficient used in loss function to obtain the error of Model-1. In deep learning, gradient-based optimizers are used to update weights and biases by using the loss of the models. Model-1 has been updated using our defined loss function by the gradient-based optimizer. Secondly, imputation is a method to complete the missing values of the dataset before the training process. Imputation was used for the prediction side for non-desired parts of reflection coefficients to decrease the loss of the interested region.

The size of the reflection coefficient produced for training is 1×1500 and the size of the binary images is 50×50. Collected reflection coefficients are between 6 GHz and 14 GHz. In binary images, black pixels represent FR4 material and white pixels represent copper. The nested-CNN was developed for 2D metamaterial, the thickness was a constant value, and the width and height could take variable values. The microwave CST simulation and Matlab program were used to collect data.

The remainder of this article is structured as follows. Section 2 describes the dataset collection and preparation. Section 3 illustrates hyperparameter optimization, the nested-CNN architecture, and prediction improvement with the imputation method. Section 4 gives models' experimental results and the margin of errors. Finally, Section 5 gives the conclusion and future work.

2. Dataset Collection and Preparation

The Microwave CST simulation program is one of the most accurate EM simulation programs, and uses various techniques to simulate the finite integration technique (FIT) [36], finite element method (FEM) [37], transmission line matrix (TLM) [38], multilevel fast multipole method (MLFMM) [39], and particle-in-cell (PIC) [40], as well as multiphysics solvers. The Microwave CST simulation program and Matlab were used to prepare the dataset. Metamaterial production and communication with the Microwave CST simulation program were made using the script written in Matlab. The dimensions of the FR4 material were fixed and its shape was determined as the cube; its dimensions were 5 cm (height), 5 cm (width), 1.6 cm (thickness). The copper shapes produced were cubes or cylinders and the size of the shape was determined according to the center point. The boundaries of the produced coppers were determined randomly, depending on the center point, not exceeding the FR4 limits. The generated coppers' width and height were variables between 0 and 5 cm depending on the origin of shape, the thickness of copper was constant and 0.035 cm, and generated coppers could not exceed FR4 limits. After all the shapes were created, the intersection regions of the copper materials were detected. Three operations were used for the intersection regions: the addition that unites two materials; the first type

of subtraction that subtracts the second material from the first material; the second type of subtraction that subtracts the first material from the second material. All operations for the intersection region are shown in Figure 1.

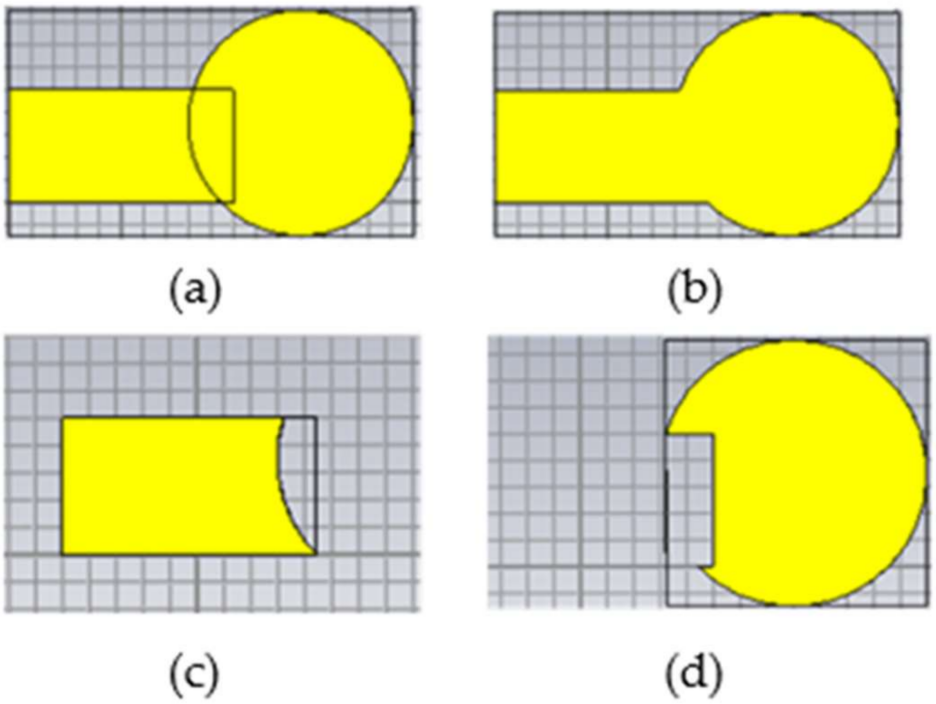

Figure 1. In (**a**), the intersection and two components; the summation operation is shown in (**b**). In (**c**), the second material is subtracted from the first. In (**d**), the first material is subtracted from the second.

For the creation of each shape, a total of x materials can be created and there is the total of y region of intersections. Based on the intersection regions, shapes can be created with the possibility of 3^y. A component can have more than one intersection region. Some other intersection regions may disappear when the subtraction is done in any order. Therefore, different shapes may occur depending on the process priority of the intersection regions. If there are y intersection regions, a shape with y! $*3^y$ probability can be produced. All possibilities were created with the Matlab script, and if any of the shapes were the same, one of the shapes was deleted to ensure the diversity of the data. The generated image of MM was 50 by 50, and the reflection coefficient's size was 1500 $\times$ 1 and was between 6 GHz to 14 GHz. The algorithm of the shape generation is shown in Algorithm 1.

The coordinate system of metamaterials is shown in Figure 2. During the simulation process, the boundary settings were chosen as given: x-axis as an open boundary, y-axis as a perfect electric wall, z-axis as a perfect magnetic wall. Two ports were used for simulation: the first one was on the positive side of the x-axis, the other one was on the negative side of the x-axis. The simulation was processed using time-domain solver with an accuracy of -40 dB.

Algorithm 1: Metamaterial's Shape Generation Process

Decide how many shapes to produce
for (# of shapes)

- decide cylindrical or cube
- decide the origin of the shape
- decide boundary conditions
- place the shape

end

for (# of intersections)
for (# of combination based on processing priority)
for (# of combination based on intersection operation for currently given processing priority)

- Simulate

end
end
end

if (shape control)

- If the shape is the same, delete one of them

end

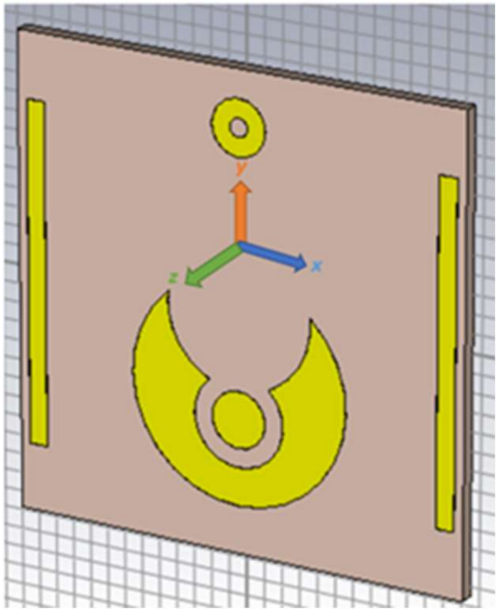

Figure 2. Metamaterial dimensions.

3. The Nested-CNN Architecture and Hyperparameter Optimization

The nested-CNN consisted of two models. Model-2 was trained to detect the reflection coefficient of the given input, which was the image of MM. Model-1 was trained to detect the image of MM of the desired reflection coefficient that was given as the input. In CNN and DNN models, weight, bias and kernels were updated with gradient-based optimization methods [41]. In gradient-based optimization methods, the model's update

process depends on the derivative of the loss value with respect to weights. The update process is given below:

$$W_t = W_{t-1} - \alpha \frac{dL}{dW_{t-1}} \quad (1)$$

where W_t is new weights, W_{t-1} is old weights, L is loss of the model, α is the learning rate. In nested-CNN, Model-2 that was used in Model-1's loss function was trained first and used in the training process of Model-1. Loss value has been created by comparing the desired reflection coefficient, which was the input of Model-1 and the reflection coefficient, which was the output of Model-2. The schematic of the nested-CNN is shown in Figure 3. Model-1's input and Model-2's output were used in the *MAE*; the desired reflection coefficient and reflection coefficient of the produced image was compared and Model-1 was updated and improved using the defined loss method. $S11_D$ is the desired reflection coefficient, $S11_P$ is the generated image's reflection coefficient, and *MAE* is calculated as given below:

$$MAE = \frac{1}{N} \sum \left| S11_D - S11_P \right| \quad (2)$$

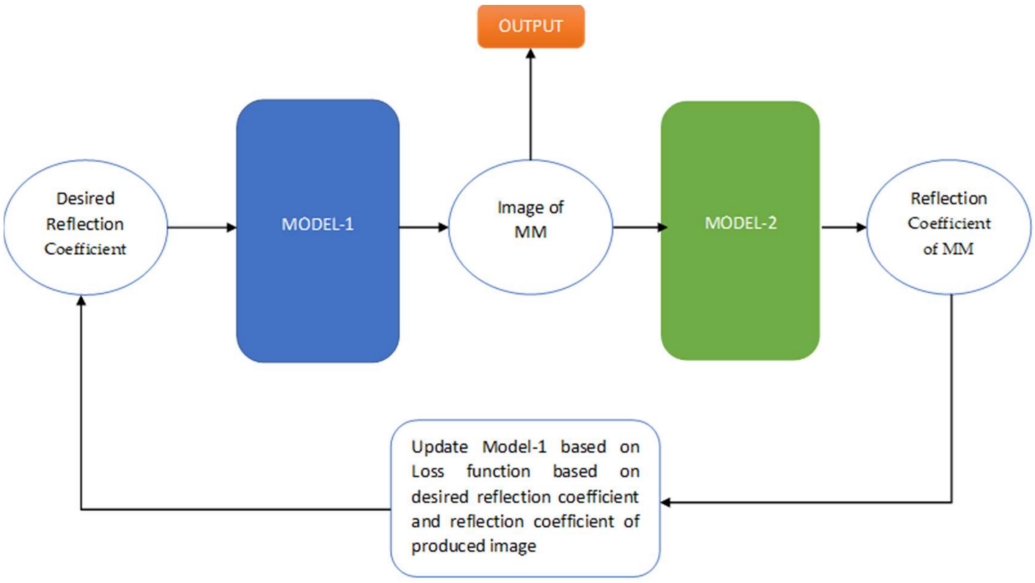

Figure 3. Schematic of the nested-CNN.

Hyperparameters are the variables that determine the artificial intelligence (AI) model structure, and the success of the CNN model depends on hyperparameters. Keras Tuner is a hyperparameter optimizer [42] that searches the parameters by using the random search algorithm [43], hyperband [44], or Bayesian optimization [45]. The random search algorithm requires more processing time than hyperband and Bayesian optimization but guarantees optimal results. In our experiment, hyperparameter optimization was provided by using Keras Tuner with the random search algorithm for both models. Parameters are given in Table 1, which were used for optimization.

Table 1. Used hyperparameters in optimization.

Hyperparameters	Values
Optimizers	Momentum/RMSprop/Adadelta/Adam
Dropout	None/0.5
Batch Size	1/2/4/8/16/32
Learning Rate	1/0.1/0.01/0.001
Hidden Layer	2 to 20, stride 1
CNN filter count	32 to 4096, stride 32
Kernel size	3 to 5, stride 1

Some popular momentum-like optimization methods, such as the root mean square error probability (RMSProp), adaptive moment estimation (Adam), and Adadelta were used to estimate the optimum direction and velocity for the cost to move towards the global minimum [46]. Dropout randomly drops layers with a specified rate to avoid overfitting; dropout was chosen—None or 0.5 for our case. The learning rate controls how quickly the model was adapted to the problem. Learning rates, large or small, have different advantages and disadvantages; smaller learning rates require a high number of epochs and there is a possibility of getting stuck before reaching the optimum result, while with larger learning rates, the result can be reached much faster, but it may converge to non-optimal results other than optimum. Four learning rates were used in hyperparameters optimizations: 1, 0.1, 0.01, 0.001. The batch size was the number of data used per iteration for training, and the batch size was investigated with values of 1, 2, 4, 8, 16, 32. CNN filters extract the feature from the portions of the image, and the kernel's size was investigated between 3 by 3 to 5 by 5. The activation function is used in deep learning models to define the output by using an input or set of inputs of the corresponding layer. The activation function mimics the stimulation of a biological neuron. In hyperparameter optimization, the used activation functions were binary, linear, sigmoid, tanh, rectified linear unit (ReLU), leaky ReLU (LReLU), parametric ReLU (PReLU), exponential linear unit (eLU), ReLU-6, scaled exponential linear unit (Selu), Softplus, Softsign and Softmax, Swish [47].

Validation and test data have been reserved to test the success of both models and to prevent overfitting. Moreso, 24,722 data was reserved for the training process, 2500 data was reserved for validation, and 2500 data was reserved for testing. Early stopping is a process that stops training when a monitored metric has stopped improving [48,49]. Early stopping was one of the key factors to find optimum weights. For Model-2, early stopping was monitoring the MAE of validation data. For the nested-CNN, early stopping was monitoring the designed loss function of validation data. Model-2 was stopped at the 63rd epoch by early stopping and the nested-CNN was stopped at the 45th epoch by early stopping.

3.1. Architecture of Model-1

Model-1's input size was 1500×1 for this situation, and one-dimensional convolutional kernels were used. Hyperparameter optimization Keras Tuner was monitoring the MAE of desired reflection coefficient and reflection coefficient of the produced image, and the optimum model is given in Table 2. Since the loss function was designed, batch size should have been 1. The Adam optimizer was selected by Keras Tuner. A dropout of 0.5 was used.

Table 2. Details and architecture of Model-1.

Model-1		
Layer	**Specifications**	**Activation Func.**
Input	1×1500	
1D-Convolution	Filters = 384, Size = (1,3), Stride = 1	Linear
1D-Convolution	Filters = 96, Size = (1,3), Stride = 1	Relu
1D-Convolution	Filters = 192, Size = (1,3), Stride = 1	Relu
1D-Convolution	Filters = 256, Size = (1,3), Stride = 1	Relu
1D-Convolution	Filters = 256, Size = (1,3), Stride = 1	Relu
Flatten		
Dense	Size = 1×2500	Sigmoid

3.2. Architecture of Model-2

In contrast to Model-1, two-dimensional convolution was used in Model-2, since the size of input was two-dimensional. Keras Tuner was monitoring the MAE of validation data, and the optimum model is given in Table 3. The batch size was 32, Adam optimizer was selected by Keras Tuner. A dropout of 0.5 was used.

Table 3. Details and architecture of Model-2.

Model-2		
Layer	**Specifications**	**Activation Func.**
Input	50×50	
2D-Convolution	Filters = 800, Size = (3,3), Stride = 1	Selu
Max Pooling	Size = (2,2)	
2D-Convolution	Filters = 192, Size = (3,3), Stride = 1	Relu
2D-Convolution	Filters =128, Size = (3,3), Stride = 1	Relu
Flatten		
Dense	Size =1500	Linear
Dense	Size =1500	Linear
Dense	Size =1500	Linear

3.3. Imputation for Prediction

Imputation was used before the training process of AI algorithms. For various reasons, many real-world datasets contain missing values, often encoded as blanks, not-a-numbers (NaNs), or other place-holders. For the usage of these datasets in AI and machine learning algorithms, imputation methods are introduced as an iterative imputer or the nearest neighbor imputation (KNN) that can replace missing values with numerical values before the training process.

The imputation method on prediction has been proposed to improve the nested-CNN's performance. Adapting the Don't Care conditions to the deep learning model in the digital system design or reducing the impact of the non-interested region during the prediction was the main goal [50]. Imputation for prediction was used to the non-interested part of the reflection coefficient to decrease the error of the interested region of the reflection coefficient. The K-nearest neighborhood method was used to impute the non-interested region of the reflection coefficient by finding the most similar signals to the desired part of the reflection coefficient. In Figure 4, the 750 samples have been selected as the non-interested part is shown.

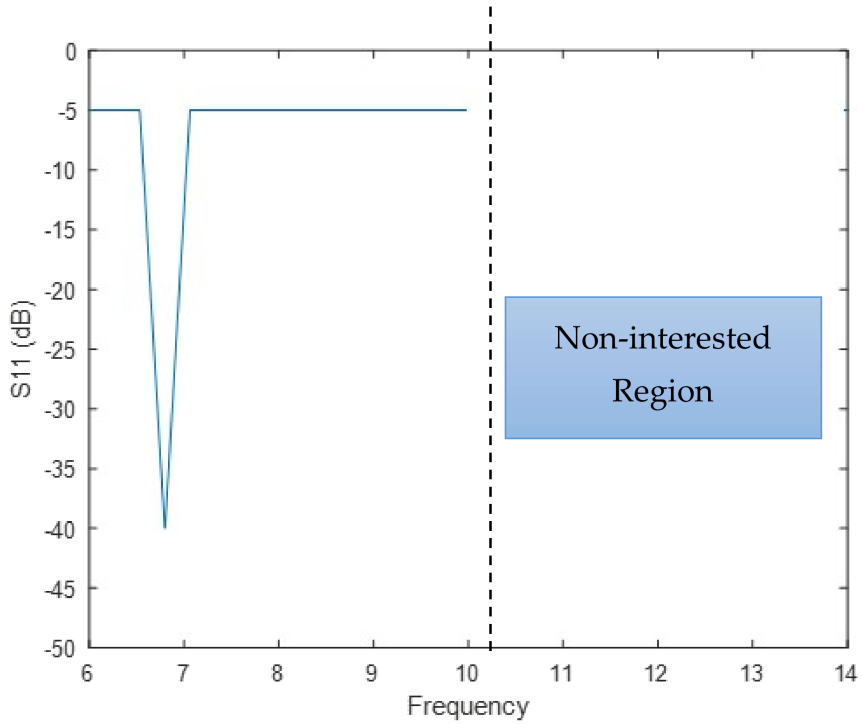

Figure 4. An example of the region of interest and the non-interested region.

KNN-imputer chooses the most similar signals to the interested region based on the Euclidian distance [51], then fills the non-interested region by using the average of the most similar neighbors. There were three factors for the KNN-imputer for the prediction side: the first one was how many samples have been used for filling, the second one was how many neighbors have been used for the KNN-imputer, and the third one was the count of the dataset. These three factors highly depend on each other. The answer to the question of how many neighbors should be used gave different results depending on the number of samples to be filled and the count of the data set. In the experiment, for the selection of the number of neighbors, it was decided that the non-interested region was used for 750 samples. The non-interested region was evaluated for the first 750 samples and the last 750 samples of each test data. The first 750 samples and the last 750 samples of each test data were filled by applying the KNN method separately. The data filled in the first part and the last part were used as separate data, thus 5000 data were created using 2500 test data, and the MAE of nested-CNN with imputation with prediction was calculated. MAE, for the first 750 samples, the last 750 samples, and the average of them, are given in the results section. The other key factor was the count of neighbors. For the last and first 750 samples, the optimum number of neighbors was searched between 10 and 200 with a stride of 10.

4. Results and Discussion

The success of Model-2 was evaluated by using MAE, the R2 score, and mean correlation coefficients for validation data and test data. The R2 score of Model-2 was 0.96. For Model-2, the MAE of training data, test data and validation data are given in Table 4.

Table 4. Mean absolute error of Model-2.

	MAE	Validation MAE	Test MAE
Model-2	0.22	0.26	0.27

The similarity between Model-2's result and the result of the simulation was measured by using correlation coefficients. The mean of the correlation coefficients was used for

validation data and test data by comparing the results of Model-2 and the simulation program. The mean correlation coefficient for test and validation data is given in Table 5.

Table 5. Mean correlation coefficient results of Model-2 and simulation based on test data.

	Mean Correlation Coefficient of the Reflection Coefficient
Test data	0.93
Validation data	0.94

Five images of MM were used to compare Model-2's result and the simulation's result. These images have never been seen by Model-2 in the training process. In Figure 5, the first column corresponds to the randomly generated image of MM, the second column corresponds to the reflection coefficients of Model-2, and the reflection coefficient of simulation. The orange line represents Model-2's result, and the blue line represents the simulation results.

Early stopping was monitoring MAE of validation for Model-2 and was stopped training at the 63rd epoch to prevent overfitting to training data. Training error and validation error for Model-2 are shown in Figure 6.

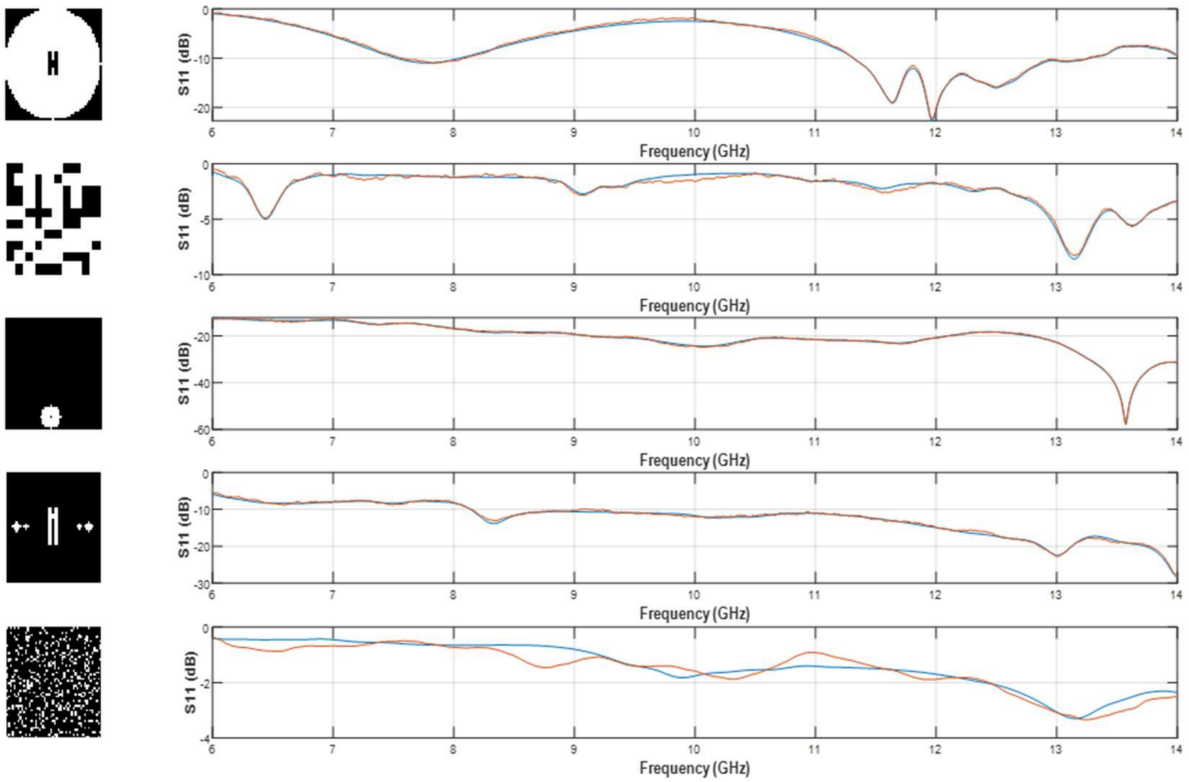

Figure 5. Five random examples of Model-2. The orange line represents Model-2's result, and the blue line represents the simulation results.

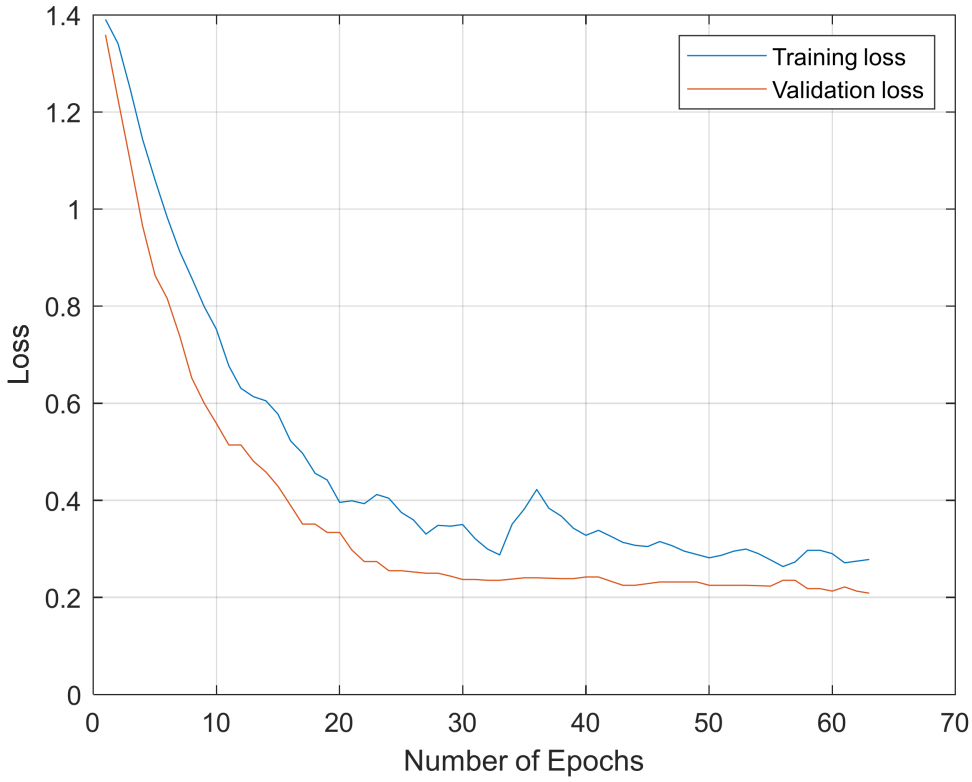

Figure 6. Training Loss and Validation Loss of Model-2.

The nested-CNN's performance was evaluated with MAE and R2 scores. The MAE of trained data, test data and validation data are given in Table 6. The R2 score of the nested-CNN was 0.9. The following situation should not be overlooked—the nested-CNN accepts the error margin of Model-2 used in the creation loss value in the training process as 0. In this case, nested-CNN's margin of error should be considered as the MAE of the nested-CNN and the sum of the MAE of Model-2.

Table 6. Mean absolute error of nested-CNN after training process, the MAE of the nested-CNN by considering the MAE of Model-2.

	MAE	Validation MAE	Test MAE
Nested-CNN	0.32	0.40	0.42
Nested-CNN + MAE of Model-2	0.54	0.66	0.69

Early stopping was monitoring the MAE of desired reflection coefficient and the reflection coefficient of the produced image of MM. The nested-CNN was stopped at the 45th epoch to prevent overfitting by early stopping. Training loss and validation loss graphics are given in Figure 7.

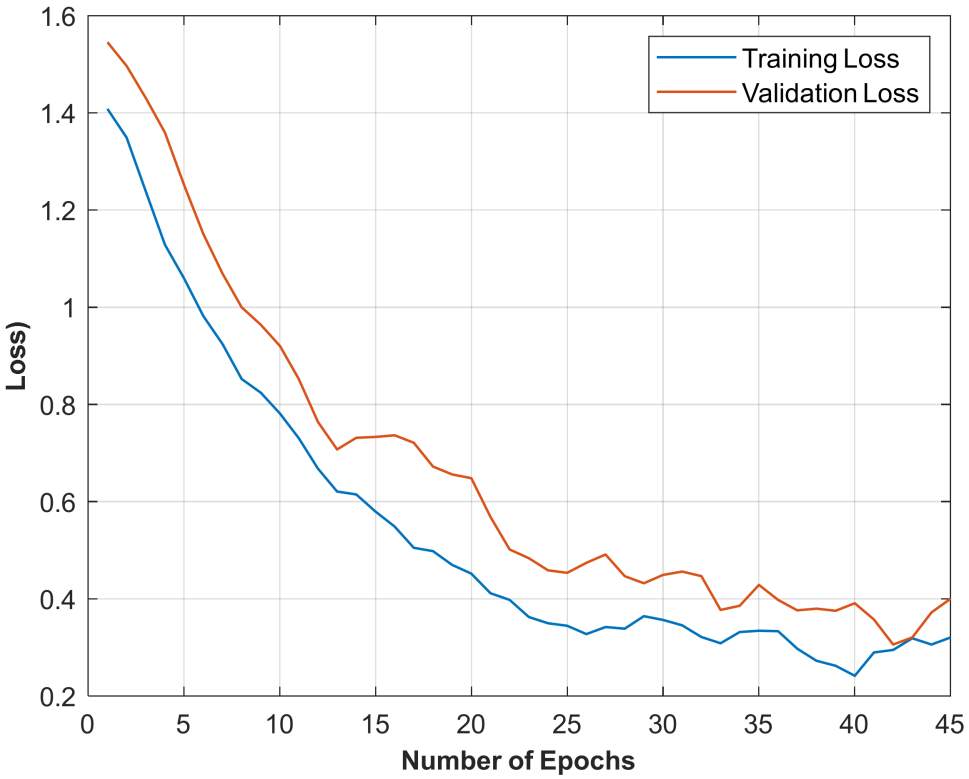

Figure 7. Training loss and validation loss of the nested-CNN.

The nested-CNN's performance was evaluated with six desired reflection coefficients. Desired reflection coefficient maximum value was chosen as −5 dB, the minimum value was chosen lower than −40 dB, the minimum bandwidth was chosen as 0.75 GHz. In Figure 8, the desired reflection coefficients, the image of MM—which is gradient-based generated by nested-CNN—and the simulation results of the image of MM are given.

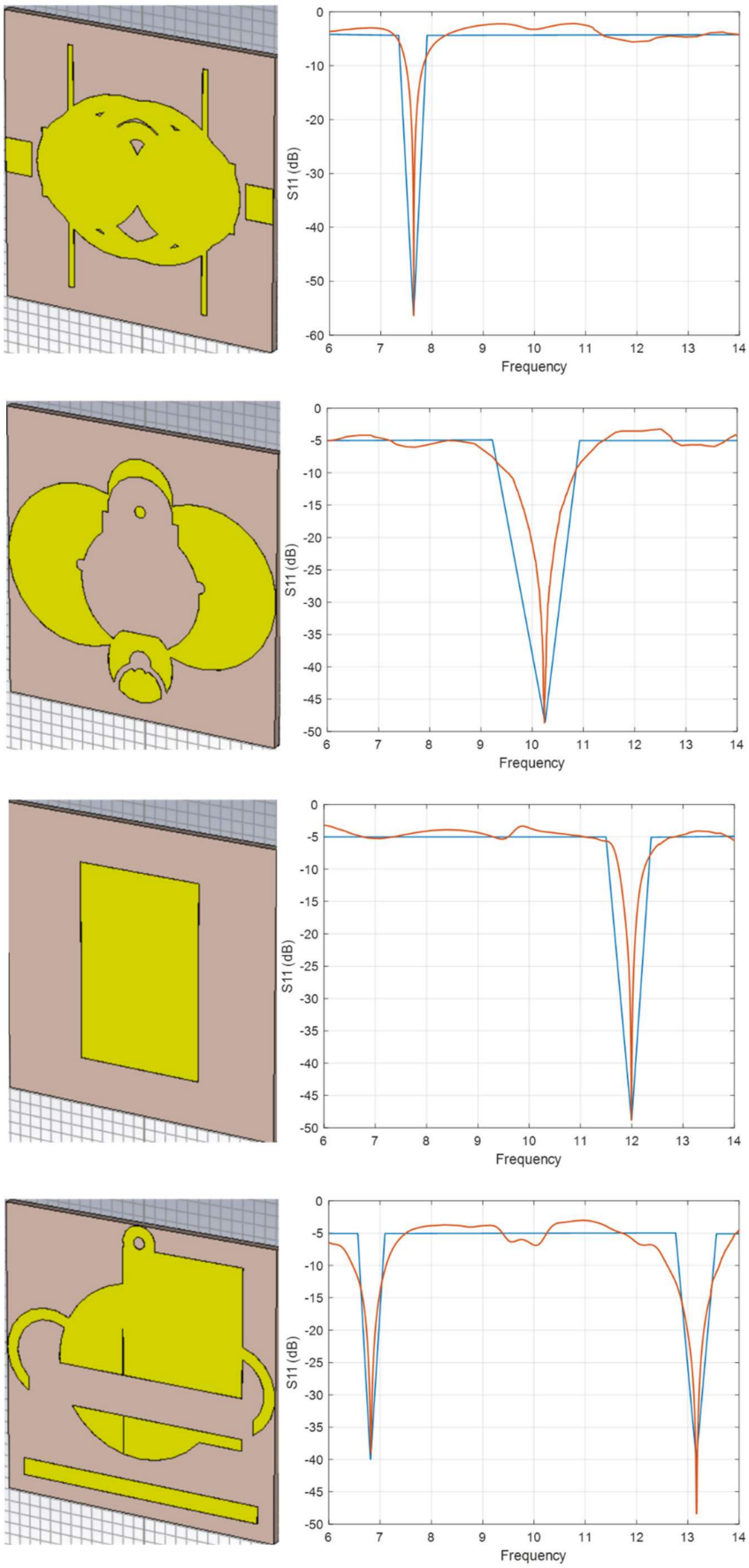

Figure 8. *Cont.*

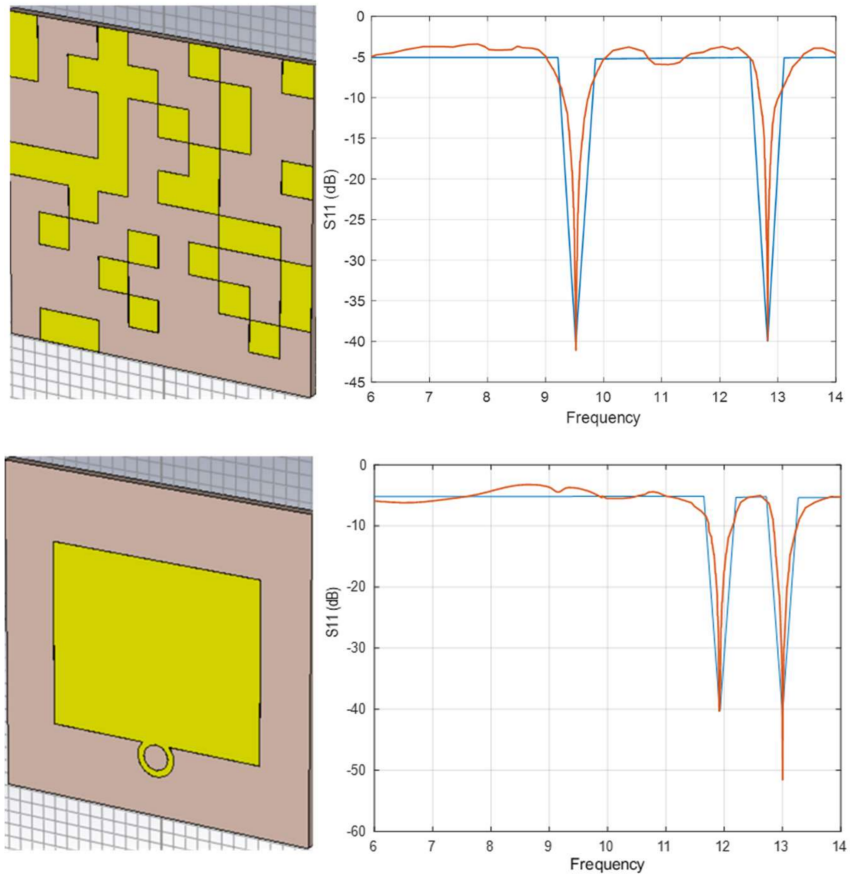

Figure 8. 6 Random examples: the blue line is desired reflection coefficients, orange line generated image by the nested-CNN's result based on simulation program.

The performance of the KNN-imputer for prediction depends on the count of the neighbors, the count of the dataset, and the samples count of the non-interested region. Moreover, the count of the neighbors depends on the samples count and the count of the dataset. In the experiment, half of the reflection coefficient, which was 750 samples, were considered as the non-interested region. The entire dataset was used except for the test data for the KNN-imputer. In these conditions, 10 to 200 neighbors were searched for test data. In total, 5000 data that filled the last end of the first part of the test data was used to obtain the MAE. In Table 7, imputed models which model with the best of two MAE scores are shown.

Table 7. MAE of nested-CNN without considering the MAE of Model-2.

	MAE of First 750 Samples Which Were Filled	MAE of the Last 750 Samples Which Were Filled	Average MAE
40 neighbors	0.16	0.19	0.18
50 neighbors	0.17	0.17	0.17

5. Conclusions

In this article, unlike traditional optimization algorithms, instead of trying the shapes with iteration to reach the desired reflection coefficient, the shape of MM was reached by giving the desired reflection coefficient to the developed nested-CNN model. To the best of our knowledge, deep learning-based models have not been applied to design MM. The nested-CNN with hyperparameter optimization was found to be promising in developing antenna and MM. It is thought that Model-2 can be used with iterative-based optimization algorithms and will provide a time advantage to optimization algorithms

made with simulation programs. By using imputation on the prediction side, the aim was to create the don't care conditions in a deep learning model, such as in digital system design. Although the effect of the non-interested region cannot be completely eliminated, its effect has been reduced. It is expected that the imputation for prediction method will provide more advantages in AI models trained in a much wider frequency range and the MAE difference will be greater. Apart from antenna and MM optimizations, it will be a new perspective, as it will provide certain advantages for multiple-input multiple-output (MIMO) systems.

In future work, component types, backplate design, thickness and frequency range expansion can be introduced to the nested-CNN model as outputs, and VSWR, permeability, permeability, scattering parameters, E-field, and H-field can be introduced to the nested-CNN model as inputs.

Author Contributions: Conceptualization, E.K.; methodology, E.K.; software, E.K.; validation, E.K. and E.E.; formal analysis, E.K. and E.E.; investigation, E.K.; resources, E.K.; data curation, E.K. and E.E.; writing—original draft preparation, E.K. and E.E.; writing—review and editing, E.E.; visualization, E.K. and E.E.; supervision, E.K. and E.E.; project administration, E.E.; funding acquisition, E.K. and E.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to currently another study is ongoing. Dataset will be shared in an appropriate repo when the ongoing study is finished.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Milias, C.; Andersen, R.B.; Lazaridis, P.I.; Zaharis, Z.D.; Muhammad, B.; Kristensen, J.T.B.; Mihovska, A.; Hermansen, D.D.S. Metamaterial-Inspired Antennas: A Review of the State of the Art and Future Design Challenges. *IEEE Access* **2021**, *9*, 89846–89865. [CrossRef]
2. Haupt, R.L.; Werner, D.H. *Genetic Algorithms in Electromagnetics*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
3. Chakravarty, S.; Mittra, R.; Fellow, L.; Williams, N.R. Application of a Microgenetic Algorithm (MGA) to the Design of Broad-Band Microwave Absorbers Using Multiple Frequency Selective Surface Screens Buried in Dielectrics. *IEEE Trans. Antennas Propag.* **2002**, *50*, 284–296. [CrossRef]
4. Chen, P.Y.; Chen, C.H.; Wang, H.; Tsai, J.H.; Ni, W.X. Synthesis design of artificial magnetic metamaterials using a genetic algorithm. *Opt. Express* **2008**, *16*, 12806–12818. [CrossRef] [PubMed]
5. Forestiere, C.; Capretti, A.; Miano, G. Data · Genetically Engineered Plasmonic Nanoarrays. *Nano Lett.* **2012**, *12*, 4, 2037–2044. [CrossRef]
6. Feichtner, T.; Selig, O.; Kiunke, M.; Hecht, B. Evolutionary Optimization of Optical Antennas. *Phys. Rev. Lett.* **2012**, *109*, 127701. [CrossRef] [PubMed]
7. Bossard, J.A.; Scarborough, C.P.; Wu, Q.; Campbell, S.D.; Werner, D.H.; Werner, P.L.; Griffiths, S.; Ketner, M. Mitigating Field Enhancement in Metasurfaces and Metamaterials for High-Power Microwave Applications. *IEEE Trans. Antennas Propag.* **2016**, *64*, 5309–5319. [CrossRef]
8. Jidi, L.; Cao, X.; Tang, Y.; Wang, S.; Zhao, Y.; Zhu, X. A New Coding Metasurface for Wideband RCS Reduction. *Radioengineering* **2018**, *27*, 394–401. [CrossRef]
9. Han, T.; Cao, X.; Gao, J.; Zhao, Y.; Zhao, Y. A Coding Metasurface with Properties of Absorption and Diffusion for RCS Reduction. *Prog. Electromagn. Res. C* **2017**, *75*, 181–191. [CrossRef]
10. Engineering, E.; Science, E. Coding metasurface for broadband microwave scattering reduction with optical transparency. *Opt. Express* **2017**, *25*, 5571–5579.
11. Yang, X. *Nature-Inspired Metaheuristic Algorithms Second Edition*; Luniver Press: Cambridge, UK, 2010.
12. Diest, K. *Numerical Methods for Metamaterial Design*; Springer: Dordrecht, The Netherlands, 2013; Volume 127, pp. 31–53.
13. Karaboga, D.; Akay, B. A comparative study of Artificial Bee Colony algorithm. *Appl. Math. Comput.* **2009**, *214*, 108–132. [CrossRef]
14. Yang, X.-S. A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 65–74.

15. Zomaya, A.Y. (Ed.) *Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2006.
16. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **1996**, *26*, 29–41. [CrossRef] [PubMed]
17. Campbell, S.D.; Zhu, D.Z.; Nagar, J.; Jenkins, R.P.; Easum, J.A.; Werner, D.H.; Werner, P.L. Inverse Design of Engineered Materials for Extreme Optical Devices. In Proceedings of the 2018 International Applied Computational Electromagnetics Society Symposium (ACES), Denver, CO, USA, 25–29 March 2018.
18. Lewis, A.; Weis, G.; Randall, M.; Galehdar, A.; Thiel, D. Antennas using Ant Colony System. 2009, pp. 1486–1492. Available online: https://ieeexplore.ieee.org/abstract/document/4983118?casa_token=De81ls4kgLIAAAAA:ZyaYSEimL-Zxkc1GsB_HDD6qrg_iAhyov0L9Fh9MDy02XZ9ifu3TL2m5FiEZj07RuJyzoxwHHwY (accessed on 9 March 2022). [CrossRef]
19. Zhu, D.Z.; Member, S.; Werner, P.L.; Member, S.; Werner, D.H. Design and Optimization of 3D Frequency Selective Surfaces Based on a Multi-Objective Lazy Ant Colony Optimization Algorithm. *IEEE Trans. Antennas Propag.* **2017**, *65*, 7137–7149. [CrossRef]
20. Zhu, D.Z.; Gregory, M.D.; Werner, P.L.; Member, S.; Werner, D.H. Fabrication and Characterization of Multi-Band Polarization Independent 3D-Printed Frequency Selective Structures with Ultra-Wide Fields of View. *IEEE Trans. Antennas Propag.* **2018**, *66*, 6096–6105. [CrossRef]
21. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
22. Robinson, J.; Rahmat-samii, Y. Particle Swarm Optimization in Electromagnetics. *IEEE Trans. Antennas Propag.* **2004**, *52*, 397–407. [CrossRef]
23. Boeringer, D.W.; Werner, D.H.; Member, S. Particle Swarm Optimization Versus Genetic Algorithms for Phased Array Synthesis. *IEEE Trans. Antennas Propag.* **2004**, *52*, 771–779. [CrossRef]
24. Cui, S.; Member, S.; Weile, D.S.; Member, S. Application of a Parallel Particle Swarm Optimization Scheme to the Design of Electromagnetic Absorbers. *IEEE Trans. Antennas Propag.* **2005**, *53*, 3616–3624. [CrossRef]
25. Jin, N.; Member, S.; Rahmat-samii, Y. Advances in Particle Swarm Optimization for Antenna. *IEEE Trans. Antennas Propag.* **2007**, *55*, 556–567. [CrossRef]
26. Ram, B. Simulated Binary Crossover for Continuous Search Space. *Complex Syst.* **1995**, *9*, 115–148.
27. Kildishev, A.V.; Chettiar, U.K.; Liu, Z.; Shalaev, V.M.; Kwon, D.; Bayraktar, Z.; Werner, D.H. Stochastic optimization of low-loss optical negative-index metamaterial. *J. Opt. Soc. Am. B* **2007**, *24*, 34–39. [CrossRef]
28. AMahmoud, K.R.; Hussein, M.; Hameed, M.F.O.; Obayya, S.S.A. Super directive Yagi–Uda nanoantennas with an ellipsoid reflector for optimal radiation emission. *JOSA B* **2017**, *34*, 2041–2049. [CrossRef]
29. Ong, J.R.; Chu, H.S.; Chen, V.H.; Zhu, A.Y.; Genevet, P. Freestanding dielectric nanohole array metasurface for mid-infrared wavelength applications. *Opt. Lett.* **2017**, *42*, 2–5. [CrossRef] [PubMed]
30. Grefenstette, J.J. Optimization of control parameters for genetic algorithms. *IEEE Trans. Syst. Man Cybern.* **1986**, *16*, 122–128. [CrossRef]
31. Kalas, A. Enhancing the particle swarm optimizer via proper. In Proceedings of the IEEE CCECE2002 Canadian Conference on Electrical and Computer Engineering, Winnipeg, MB, Canada, 12–15 May 2002; pp. 792–797.
32. Li, C.; Yang, S.; Nguyen, T.T. A Self-Learning Particle Swarm Optimizer for Global Optimization Problems. *IEEE Trans. Syst. Man Cybern. Part B* **2012**, *42*, 627–646. [CrossRef]
33. Hansen, N.; Müller, S.D.; Koumoutsakos, P. Reducing the Time Complexity of the Derandomized Evolution Strategy. *Evol. Comput.* **2003**, *11*, 1–18. [CrossRef] [PubMed]
34. Sieber, P.E.; Werner, D.H. Infrared broadband quarter-wave and half-wave plates synthesized from anisotropic Bézier metasurfaces. *Opt. Express* **2014**, *22*, 32371–32383. [CrossRef] [PubMed]
35. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaría, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *J. Big Data* **2021**, *8*, 1–74. [CrossRef]
36. Munteanu, I.; Weiland, T. RF & Microwave Simulation with the Finite Integration Technique—From Component to System Design. In *Scientific Computing in Electrical Engineering*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 247–260. [CrossRef]
37. Augustyniak, M.; Usarek, Z. Finite Element Method Applied in Electromagnetic NDTE: A Review. *J. Nondestruct. Eval.* **2016**, *35*, 1–15. [CrossRef]
38. Hoefer, W.J.R. The Transmission-Line Matrix Method—Theory and Applications. *IEEE Trans. Microw. Theory Tech.* **1985**, *33*, 882–893. [CrossRef]
39. Darve, E. The Fast Multipole Method: Numerical Implementation. *J. Comput. Phys.* **2000**, *160*, 195–240. [CrossRef]
40. Meierbachtol, C.S.; Greenwood, A.D.; Verboncoeur, J.P.; Shanker, B. Conformal Electromagnetic Particle in Cell: A Review. *IEEE Trans. Plasma Sci.* **2015**, *43*, 3778–3793. [CrossRef]
41. Dogo, E.M.; Afolabi, O.J.; Nwulu, N.I.; Twala, B.; Aigbavboa, C.O.; Science, E.E.; Survey, Q.; Africa, S. A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks. In Proceedings of the 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), Belgaum, India, 21–22 December 2018; pp. 92–99.

42. Pon, M.Z.A.; KK, K.P. Hyperparameter Tuning of Deep learning Models in Keras. *Sparklinglight Trans. Artif. Intell. Quantum Comput.* **2021**, *1*, 36–40.
43. Zabinsky, Z.B. *Random Search Algorithms*; Department of Industrial and Systems Engineering, University of Washington: Washington, DC, USA, 2009.
44. Li, L.; Jamieson, K.; Desalvo, G. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *J. Mach. Learn. Res.* **2018**, *18*, 6765–6816.
45. Pelikan, M.; Goldberg, D.E.; Cantú-Paz, E. BOA: The Bayesian optimization algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, Orlando, FL, USA, 13–17 July 1999; pp. 525–532.
46. Zou, F.; Shen, L.; Jie, Z.; Zhang, W.; Liu, W. A Sufficient Condition for Convergences of Adam and RMSProp. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 11127–11135.
47. Szandała, T. Review and comparison of commonly used activation functions for deep neural networks. *Stud. Comput. Intell.* **2021**, *903*, 203–224. [CrossRef]
48. Prechelt, L. Early Stopping—But When? In *Neural Networks: Tricks of the Trade*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 55–69. [CrossRef]
49. Yao, Y.; Rosasco, L.; Caponnetto, A. On Early Stopping in Gradient Descent Learning. *Constr. Approx.* **2007**, *26*, 289–315. [CrossRef]
50. Damiani, M.; De Micheli, G. Don't Care Set Specifications in Combinational and Synchronous Logic Circuits. *IEEE Trans. Comput. Des. Integr. Circuits Syst.* **1993**, *12*, 365–388. [CrossRef]
51. Donders, A.R.T.; van der Heijden, G.J.M.G.; Stijnen, T.; Moons, K.G.M. Review: A gentle introduction to imputation of missing values. *J. Clin. Epidemiol.* **2006**, *59*, 1087–1091. [CrossRef] [PubMed]