B. TECH. PROJECT REPORT

On

Microwave Absorbers

BY

Khush Bachara(180002028)
Akhil Atri (180002004)

DISCIPLINE OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
May 2022

Microwave Absorbers

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degrees

of
BACHELOR OF TECHNOLOGY
in

ELECTRICAL ENGINEERING

Submitted by:
Khush Bachara(180002028)
Akhil Atri(180002004)

Guided by:
DR. SAPTARSHI Ghosh

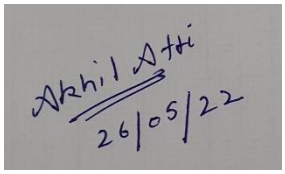
INDIAN INSTITUTE OF TECHNOLOGY INDORE
MAY 2022

CANDIDATE'S DECLARATION

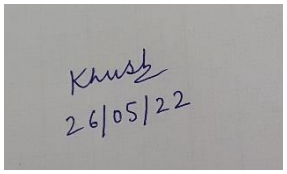
We hereby declare that the project entitled **"Microwave Absorbers"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Electrical Engineering' completed under the supervision of **Dr. Saptarshi, Professor,** IIT Indore is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

Akhil Atri



Khush Bachara



Signature and name of the student(s) with date

–

CERTIFICATE by BTP Guide(s)

It is certified that the above statement made by the students is correct to the best of my/our knowledge.

Saptarshi Ghosh 26/05/2022

Signature of BTP Guide(s) with dates and their designation

Dr. Saptarshi Ghosh, Assistant Professor, EE

Preface

This report on "Microwave Absorbers" is prepared under the guidance of Dr. Saptarshi Ghosh.

(Through this report we have tried to give a detailed information of a microwave absorber from which we have obtained the optimized parameters using neural networks.

We have tried to the best of our abilities and knowledge to explain the content in a lucid manner. We have also added models and figures to make it more illustrative.)

Akhil Atri and Khush Bachara
B.Tech. IV Year
Discipline of Electrical
IIT Indore

Acknowledgements

We wish to thank Dr. Sapatrshi Ghosh for his kind support and valuable guidance.

It is his help and support, due to which we became able to complete our project.

Without his support this report would not have been possible.

Akhil Atri(180002004)
Khush Bachara(180002028)
B.Tech. IV Year
Discipline of Electrical Engineering
IIT Indore

Abstract

This research presents a novel polarization-insensitive, angular stable, tunable fractal frequency selective surface (FSS)-based GigaHertz (GHz) multiband and dualband absorber. A Neural network (NN) approach based is proposed and used to optimise the absorber to the desired frequency and bandwidth. The smallest mean square error for multiband absorber between the target and observed outputs is 0.015214603, and for the boradband absorber the smallest mean square error between the target and observed outputs is 0.0004774. Hence more close the mean square error is to the zero, more better the NN model is. At a central frequency of 10.5 GHz, maximum absorption of 62 percent is achieved for multiband absorber. Where as at a central frequency of 10.5 GHz, maximum absorption of 85 percent is achieved for broadband absorber. Furthermore, the designed absorber works well for both perpendicular and parallel polarizations with incidence angles up to 45^0. The results from the NN were confirmed with the ansys software, and there was around 98% accuracy in NN to predict the frequency. THz imaging, sensing, detection, and stealth are all possible applications for the proposed absorber.

Table of Content

List of Figures

List of Tables

CHAPTER 01: INTRODUCTION

BACKGROUND

Absorbers have been utilised in radar cross-section reduction, EMI suppression, and sensor applications. A perfect absorbing structure with perfect impedance matching has no transmission/reflection at a working frequency. Good absorption, a high absorption bandwidth, a thin substrate, polarisation insensitivity, and a wide angular stability are all characteristics of an efficient absorber. However, meeting all of these requirements at the same time is a difficult endeavour. The FSS-based absorbers are the most effective at simultaneously satisfying all of the above conditions Frequency selective surfaces (FSSs) are multiband electromagnetic filtering periodic structures that operate at microwave and terahertz frequencies. GHz absorbers with FSS support are those that can give a wide absorption bandwidth while remaining thin. These structures have a wide range of angular stability and polarisation insensitivity. On the backside of an FSS-based GHz absorber, a substrate separates a periodic resonating structure and a reflector. By matching the impedance to the free space impedance at specific frequencies, this surface minimises reflection.

The optimum parameters for a GHz absorber with a 1.5 GHz absorption bandwidth were obtained using an optimization technique. Using a machine learning approach to analyse and optimise GHz absorbers has been unusual in the literature so far. The main objective of this thesis is to present an effective neural network strategy for realising polarization-insensitive, angular stable GHz absorbers which is thoroughly backed and validated by Ansys Hfss software findings. The following is the work's structure: The analytical process for building a microwave absorber is demonstrated in Section IV of the thesis. Chapters V and VI cover neural network training using simulated data as well as structural analysis. Section VII discusses the findings and conclusions.

MOTIVATION

Metamaterials (MMs) are materials with particular electromagnetic properties, such as a negative refractive index, left-hand behaviour, remarkable transmission, negative Doppler effect, and so on. Applications of MMs have gotten a lot of interest in recent years because of their unique features. Several devices, such as the perfect lens, invisibility cloaking, perfect absorber, and transmission, have been conceived and constructed for practical application in the GHz band-based system. So, using NN, this study recommends the optimal parameters for the absorber, which will aid in the selection of absorbers based on requirements.

Chapter 02: Introduction about absorbers

Absorbers

A periodic ordered metallic pattern layer, a dielectric layer, and a continuous metallic layer make up the bulk of Metamaterial Perfect Absorbers. Each of MMPA's tiers has its own set of abilities. The first layer equalises the impedance mismatch between the absorber and the environment (air). The incident field is allowed to remain and be absorbed by the second layer. Finally, in order for the field that was unable to be absorbed in the middle layer to be absorbed again, the third layer must reflect it.

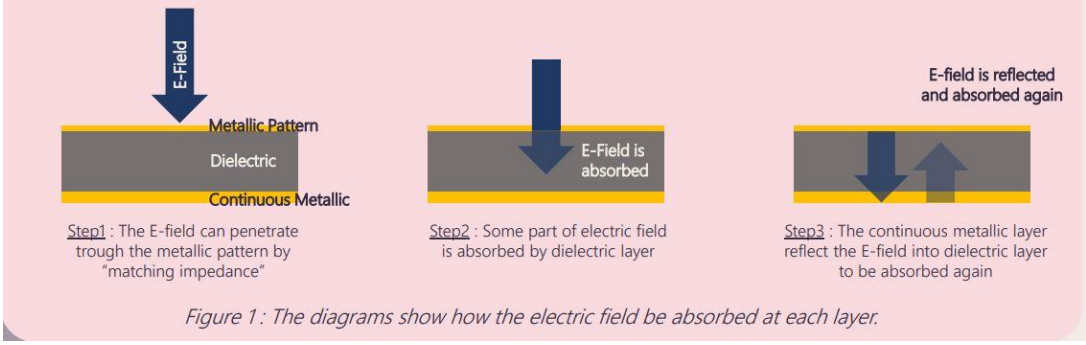


Figure 2.1 : Absorption of electric field at different layer

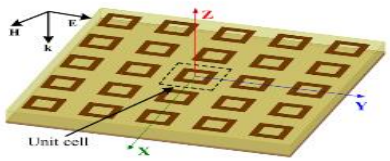


Figure1.2: Hfss cell Structure

- A microwave absorber exhibits absorption at microwave frequency(ies) : Single/Multiband absorber.
- Absorptivity (A) is greatest when reflected power (|S11|2) and transmitted power (|S21|2) are both decreased concurrently.

$$A = 1 - |S_{11}|^2 - |S_{21}|^2$$

- Most of the absorber designs are backed by complete metal lamination, thereby ensuring zero transmission (i.e. $|S_{21}|=0$).
- Then, the absorptivity equation reduces to : $A = 1 - |S_{11}|^2$

Because of the design, the input impedance is ideally matched to the free space impedance. (when $|S_{21}|=0$)

$$Z_{in} = \eta_0 \sqrt{\frac{(1+S_{11})^2 - S_{21}^2}{(1-S_{11})^2 - S_{21}^2}} = \eta_0 \frac{1+S_{11}}{1-S_{11}} \Longrightarrow S_{11} = \frac{Z_{in} - \eta_0}{Z_{in} + \eta_0}$$

Impedance Matching Criteria

The absorber structure based on FSS reduces both.: S_{11} and S_2

CHAPTER 03: NEURAL NETWORKS USING PYRENN

PYRENN :- Pyrenn is a Python-based recurrent neural network toolkit .It is developed completely in Python and Numpy, and it lets you build a wide range of (recurrent) neural networks. It's easy to use and is well-documented.

Advantages of pyrenn:-

- Pyrenn allows you to design a wide range of (recurrent) neural network configurations.
- Neural networks are simple to construct, train, and use with the help of pyrenn.
- Instead of first-order approaches like gradient descent, it trains using the Levenberg-Marquardt algorithm (a second-order Quasi-Newton optimization approach).
- Only python and numpy are used in the python version
- The algorithms Real-Time Recurrent Learning (RTRL) and Backpropagation Through Time (BPTT) have been created and may be utilised to generate new training techniques.

CREATING A NEURAL NETWORK USING PYRENN

Using the CreateNN function, a pyrenn neural network object that is capable of being trained and utilised is created. When only the short notation nn is provided as input, the neural network that is built will be an MLP and will not have any delayed connections. If a (recurrent) neural network with delays is to be formed, the parameters dIn, dIntern, and/or dOut must be specified in the same manner as was previously stated.

In Python :- Pyrenn.CreateNN(nn, [dln = [0], dintern = [], dOut = []])

This function creates an object representing a neural network with weights having random values in the range -0.5 to 0.5.

Table 3.1: Attributes in the command of creating a neural network in pyrenn..

Attribute	Value
Nn(list)	is a short form of the neural network

dln(list)	is a set that represents the input delays of the neural network.
dintern(list)	is a collection of the inernal delays that the neural network possesses.
dOut(list)	is the set of delays that are generated by the neural network's output.

TRAINING OF A NEURAL NETWORK

A neural network may be taught once it has been built. You'll need training data to build a neural network. When given the input p and the output y, (p,y) represents one sample of training data.

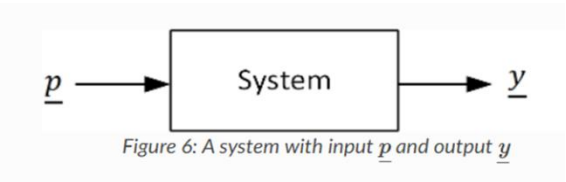
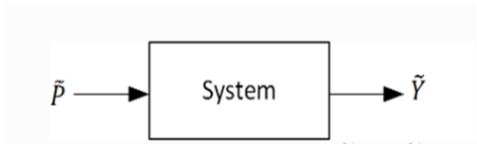


Figure 6: A system with input $\underline{p}$ and output $\underline{y}$

Figure 2.1 : A system with input p and output y

When training neural networks, more than one data sample is usually required to produce effective results. As a result, an input matrix and an output (or target) matrix include Q training data samples define the training data. Only static systems (feed forward neural networks) need that the input matrix element q equal the output matrix element q. (in any given order). In dynamic systems (recurrent neural networks), the samples must be in the right order. Both systems' training data should be as accurate as feasible.



$$\tilde{P} = \begin{bmatrix} \underline{p}[1] & \underline{p}[2] & \cdots & \underline{p}[q] & \cdots & \underline{p}[Q] \end{bmatrix}$$
$$\tilde{Y} = \begin{bmatrix} \underline{y}[1] & \underline{y}[2] & \cdots & \underline{y}[q] & \cdots & \underline{y}[Q] \end{bmatrix}$$

The neural network may be trained using the training data. The weights in the weight vector w must be gradually changed to ensure that the neural network output matches the training data output while training a neural network (target). This optimization aims to decrease the E (cost function) error between neural network and system outputs.

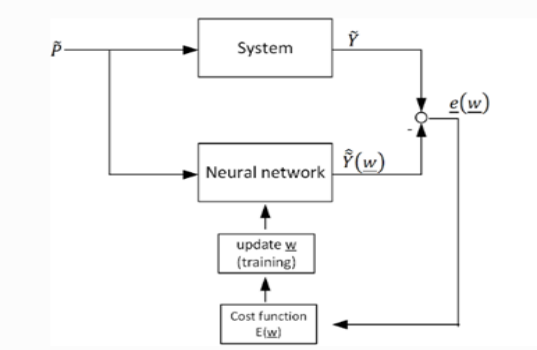


Figure 3.2: Training loop of Neural Network

Note :- When it comes to neural network training, the error E, also known as the cost function, can be computed in a variety of methods. Pyrenn uses the mean squared error in every calculation, which is essential when utilising the Levenberg-Marquardt method.

Until one of two termination criteria is reached, the training will continue to adjust the weights of the weight vector w:

- when the maximum number of repetitions (epochs) kmax, has been reached
- the error has been reduced to the desired level of E <Estop.

train_LM() : train with Levenberg-Marquardt Algorithm

The train LM() function is an implementation of the Levenberg-Marquardt algorithm (LM) based on:

The LM algorithm is a second-order optimization technique that uses the Jacobian matrix J to approximate the Hessian matrix H. Pyrenn uses the Real-Time Recurrent Learning (RTRL) technique to generate the Jacobian matrix.

pyrenn.train_LM(P, Y, net[, kmax=100, E stop=1e-10, dampfac=3.0, dampconst=10.0, verbose=False])

Using the training data inputs P and outputs (targets) Y, the Levenberg-Marquardt method is used to train the given neural network.

Parameters :

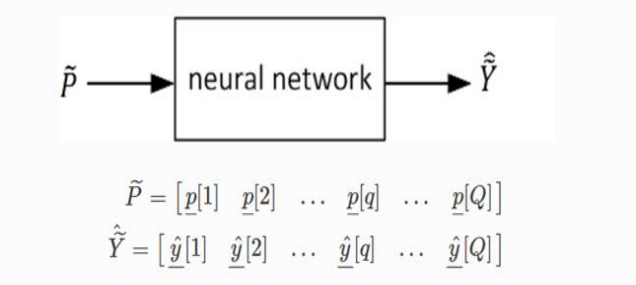
Table 3.2 : Attributes in the command of training a neural network in pyrenn..

Parameters	Value
P(numpy.array)	P, 2d-array of form (R,Q) with R rows (=number of inputs) and Q columns (=number of training samples) for training input data.
Y(numpy.array)	Y, 2d-array of shape (SM,Q) containing SM rows (=number of ouputs) and Q columns (=number of training samples) for the training ouput (target) dataset.
Net(dict)	pyrenn produced this neural network object. CreateNN().
K_max(int)	number of training iterations maximum (epochs)
Dampfac(float)	The LM algorithm's damping factor.
Dampconst(float)	constant to adjust LM damping factor

Verbose	If True, triggers console outputs during training.

USING A TRAINED NEURAL NETWORK.

After a neural network has been correctly trained, it can be used to calculate neural network outputs for fresh input data that is not the same as that used for training. The input data known as has the same structure as when the neural network was trained when it is utilised. The structure of the computed output data generated by the neural network is identical to that of the training output data. It is feasible to use any arbitrary number of data samples, represented by the letter Q, to produce the same number of output samples.



Saving and loading of a neural network

The saveNN() function can be used to save the training weights of a neural network as well as the network's topology in a csv file. A previously saved neural network can be retrieved using the loadNN function. This capability allows users of Python and Matlab to change out their neural network objects.

Save neural network with :- saveNN()

Python function :-Pyrenn.saveNN(net,filename)

The above python function saves a neural network object to a csv file

Parameters for the Pyrenn.saveNN() function :-

Table 3.3 : Attributes in the command of training a neural network in pyrenn..

Parameter	Value
net (dict)	This item represents a pyrenn neural network.
filename (string)	is the complete or relative path to the neural network's csv file (folderfile.csv).

CHAPTER 04: Structure and Simulated Data

The unit cell architecture for the recommended design is shown in Figure 2, and it comprises of three stacked dielectric layers with cross-dipole metallic patches imprinted on the top surface of each substrate. A continuous metal plane supports the whole structure (made of three metal-dielectric stacked layers) to achieve zero transmission. The dielectric layer has a thickness of 0.2 mm and is made of FR4 (r = 4.4 and tan = 0.02). The metallic patterns and the ground plane are made of copper (= 5.8 107 S/m), with a thickness of 0.018 mm for each. Depending on the absorber type, the cross-dipole patterns have identical forms but varied geometric proportions.

A resonance frequency will be generated by each cross-dipole etched on the dielectric.The resonance frequencies are variable due to the varying diameters of the cross-dipoles. As a consequence, the resonance frequencies corresponding to each metal-dielectric layer may be adjusted by optimising the geometric dimensions, allowing the same structure to function as a triple-band absorber and a 10 dB bandwidth-enhanced absorber.

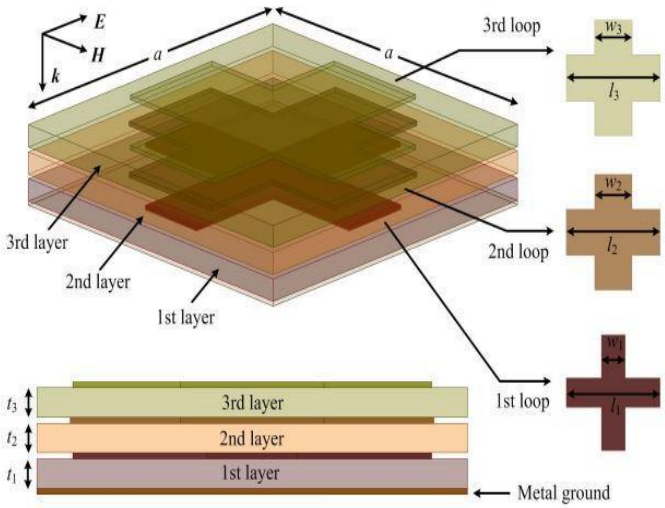


Figure 4.1: Structure of Multilayer Absorber

Software used for simulations

For simulation of data and structure to get the parameters, we used the ansys software.

ANSYS :-

ANSYS HFSS is a 3D electromagnetic (EM) simulation tool for radio wave components, antenna rays, high-speed interconnects, connectors, IC packages, printed circuit boards, and microwave components. Engineers use ANSYS HFSS to build and simulate high-speed, high-frequency electronics in radar systems, communication systems, satellites, ADAS, IoT items, and other high-speed RF devices all around the world.

Larger structures are often broken down into smaller components that are independently modelled and tested in Ansys. The weight, pressure, temperature, and other physical attributes of an item can be defined first, followed by its dimensions. Finally, the Ansys programme simulates and analyses over time movement, fatigue, fractures, fluid flow, temperature distribution, electromagnetic efficiency, and other effects.

HFSS is a commercial tool for designing antennas and complex radio frequency electrical circuit components such filters, transmission lines, and packaging.

SIMULATED DATA

The property attributes on which our structure was getting simulated are as follows :-

Table 4.1 : Attributes of absorber structure

Solution Frequency	Single
Frequency	5 GHz
Maximum number of passes	15
Maximum Delta S	0.02
Sweep Name	Sweep
Sweep Type	Interpolating

Dataset for Broadband/Multiband absorber

Input dataset :- For an input dataset we considered 9 different parameters by performing simulations on using the ansys software which are as follows :-

Table 4.2 : Independent parameters of the absorber structure

Parameters	A	b	l1	w1	l2	w2	l3	w3	T

Output dataset :- For an output dataset we considered -10db as our target variable which is the parameters we got in result by performing various simulations on the input parameters.

Table 4.3 : Dependent parameter of the absorber structure

Parameter	-10dB (f)

Significance of all the parameters considered for our dataset.

Table 4.4 : Explanation of the parameters considered for the absorber.

Parameter	Significance
a	Length of absorber structure
B	Breadth of absorber structure
l1	Length of loop 1
w1	Width of loop 1
l2	Length of loop 2
w2	Width of loop 2
l3	Length of loop 3
w3	Width loop 3
T	Thickness of substrate
-10dB (f)	Resonant frequency

Let us have a look at our training dataset. Here are few training datapoints:-

Table 4.5 : Sample of dataset from simulation via ansys software.

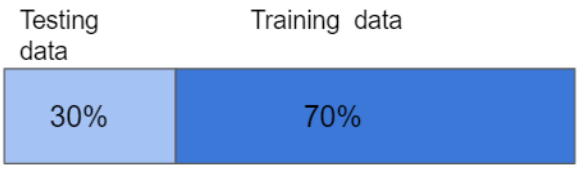
A	b	L1	W1	L2	W2	L3	W3	t	-10dB(f)
9	9	8.9	2.6	8.2	3	6.7	3	0.2	0.36
9.2	9.2	8.9	2.6	8.2	3	6.7	3	0.2	0.37
9.1	9.1	8.9	2.4	8.3	3.2	6.7	3	0.2	0.408

Splitting of dataset.

For the training and testing of neural network using LM algorithm :-

- 70% of the data was utilised for training purposes.

- 30% of the data was used for testing purposes.



CHAPTER 05: Training of Neural Network

Q:- HOW DID WE TRAINED THE NEURAL NETWORK ?

- We used Levenberg-Marquardt algorithm.
- The LM algorithm is a second-order optimization technique that uses the Jacobian matrix J to approximate the Hessian matrix H. The Real-Time Recurrent Learning (RTRL) technique is used to construct the Jacobian matrix in Pyrenn.
- Using the training data inputs P and outputs (targets) Y, the Levenberg-Marquardt method is used to train the given neural network net.
- pyrenn.train LM = Python function (P, Y, net[, k max=100, E stop=1e-10, dampfac=3.0, dampconst=10.0, verbose=False], verbose=False]

LM ALGORITHM

The LM method was created to deal with loss functions that were expressed as a sum of squared errors. It works even if the precise Hessian matrix isn't determined. Instead, the Jacobian matrix and gradient vector are employed. Consider a loss function defined as the sum of squared errors.

f=m∑i=1e2if=∑i=1mei2

The number of training samples is given by mm. The derivatives of the mistakes affecting the parameters are contained in the Jacobian matrix of the loss function.

for $i=1,\ldots,m_i=1,\ldots,m$ and $j=1,\ldots,n_j=1,\ldots,n$.

The data set has mm samples, and the number of neural network parameters is nn. The Jacobian matrix is mnmn bytes in size. The gradient vector of the loss function can be calculated as follows:

∇f=2JT·e∇f=2JT·e

Here ee is the vector of all error terms. Finally, we may use the following expression to

approximate the Hessian matrix.

Hf≈2JT·J+λIHf≈2JT·J+λI

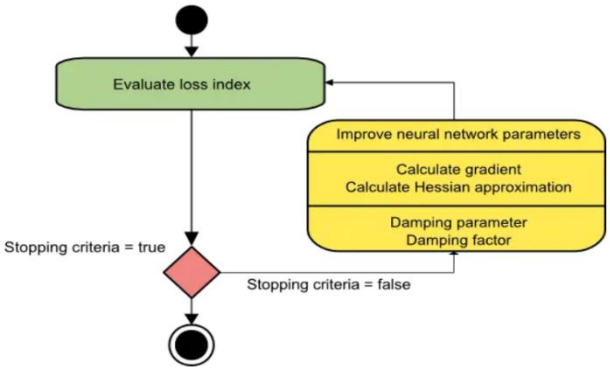
Where is the identity matrix and is a damping factor that ensures the Hessian is positive. The Levenberg-Marquardt algorithm is used to optimise the parameters in the following statement.

w(i+1)=w(i)−(J(i)T·J(i)+λ(i)I)−1·(2J(i)T·e(i)),w(i+1)=w(i)−(J(i)T·J(i)+λ(i)I)−1·(2J(i)T·e(i)),

for i=0,1,…i=0,1,….

When the damping value is zero, Newton's technique is used, which uses an approximation Hessian matrix. When is large, however, this becomes gradient descent with a slow training rate.To ensure that the first updates are tiny steps in the direction of gradient descent, the parameter is set to a large value. If each iteration fails, the value is raised by some amount. The Levenberg-Marquardt method approaches the Newton technique if is lowered as the loss decreases. This method frequently speeds up the convergence to the minimum.

The Levenberg-Marquardt method is used to train a neural network, as seen in the state diagram below. To begin, compute the loss, gradient, and Hessian approximation. To minimise the loss, the damping parameter is changed for each iteration.



Why did we used LM algorithm for training?

The optimization algorithm manages the learning process in a neural network (or optimizer).

Various optimization algorithms exist. But memory needs, processor speeds, and numerical precision vary a lot for each one of them. Below are the various training algorithms:

• Descent Gradient

• Newton algorithm

• Conjugate gradient

• Quasi-Newton algorithm

• Levenberg-Marquardt algorithm

Performance comparison of different training algorithms

The chart below depicts the training algorithms' computational speed and memory requirements.

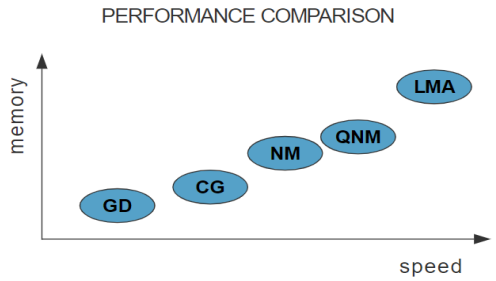


Figure 5.1: Performance Comparison of Training Algorithm

- Gradient descent is the slowest training algorithm, but it also uses the least amount of memory, as we can see.
- On the other hand, the Levenberg-Marquardt algorithm is usually the fastest, although it consumes a lot of memory.
- The quasi-Newton approach might be a nice middle ground.

Final selection of training algorithm

To summarise, we can conserve memory by employing gradient descent or conjugate gradient if our neural network has thousands of parameters. If we have a large number of neural networks to train with only a few thousand samples and a few hundred parameters, the Levenberg-Marquardt technique may be the best alternative. The quasi-Newton approach will suffice in all other circumstances.

Prioritization :- Finally, we had no memory issues with our research because we used

Google Colab, a cloud environment, to construct and deploy our models, and we did not have a large data collection. The LM algorithm was the obvious choice for us.

Neural Network Architecture:-

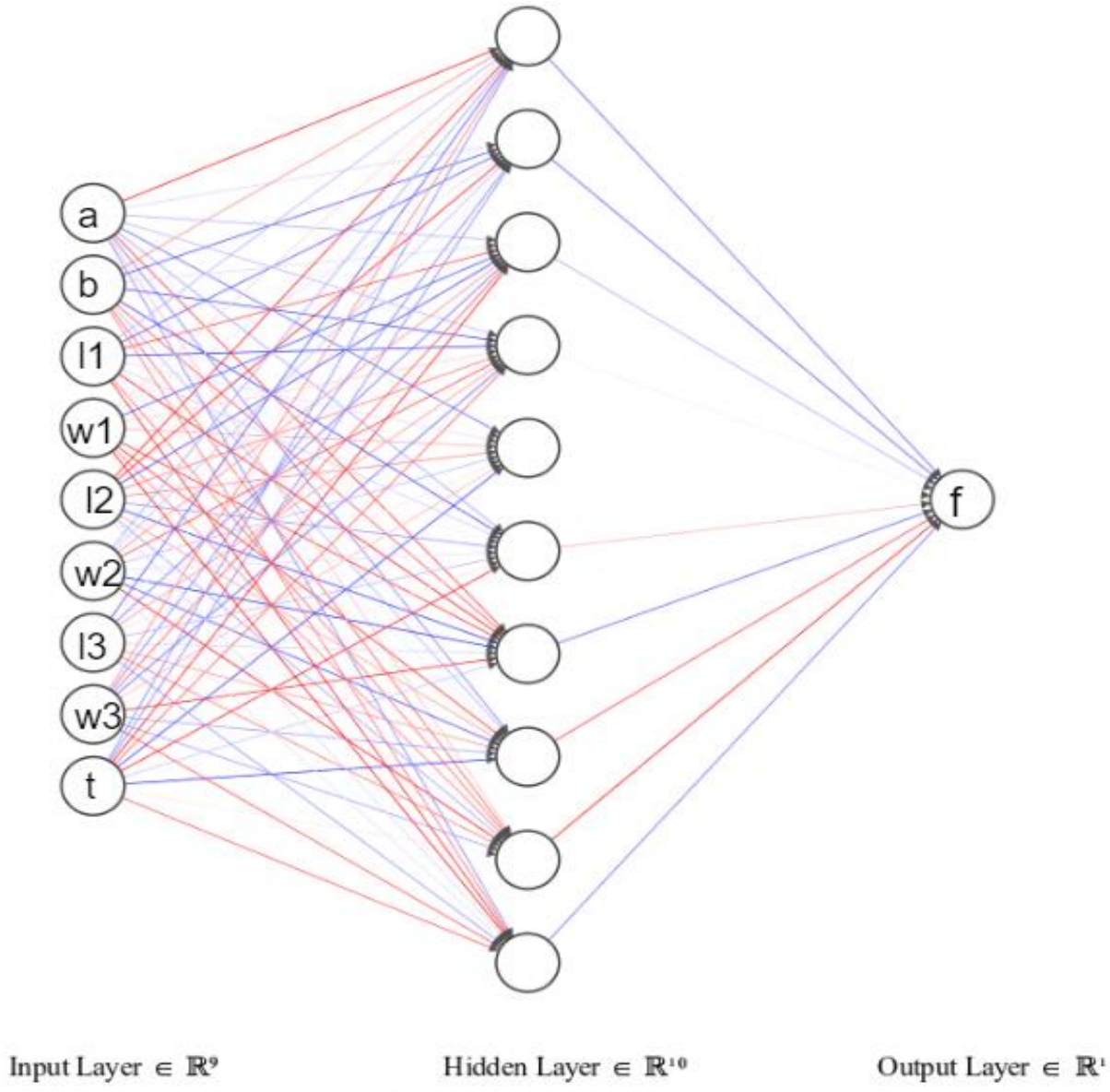


Figure 5.2 : Representing graphical view of neural network architecture

The below mentioned table represents the paramaters and their associated values which have been used to create the neural network.

Parameters	Values
Number of input layer	1
Number of hidden layer	1
Number of output layer	1
Inputs given to input layer	9
Neurons in hidden layer	10
Activation used in hidden layer	Hyperbolic tangent
Activation used in output layer	Linear
Training algorithm	Levenberg-Marquardt (LM) algorithm
Iterations	100

Table 5.1 : Parameters with values used to create the neural network.

CHAPTER 06: Analysis

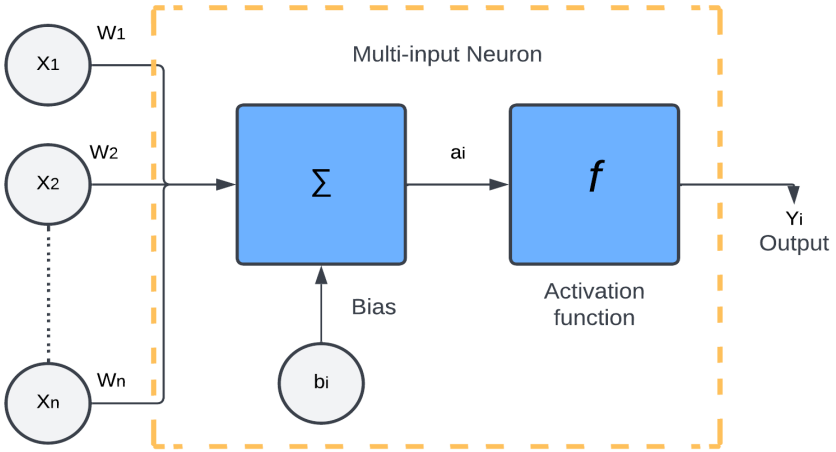


Figure 5a

Figure 5a depicts the fundamental anatomy of a neuron. Each link between nodes (neurons) is assigned a weight. To normalise the output, the neural network employs an activation function. The expression may be used to determine the output of a neuron.

$$Y_i = f(\sum x_i \, w_{ji} + b_j \;)$$

where f is the activation function, b is the bias, w is the weight of the node connections, and x and y are the input and output. The architecture of the NN used in this project is shown in Figure 7a. It has ten neurons in the hidden layer, nine absorber parameters in the input layer, and one nodes for resonance frequency in the output layer. The model proposed in the previous section is utilised to create a database of 160 records which was exctracted by performing more than 200 number of simulations on ansys software. After the data is generated, the neural network is trained using Levenberg-(LM) Marquardt's method. To update the weight of the linkages between the nodes, LM employs the expression: command.

$$\text{delta w} = (J^{'}J + \mu I)J^{'}e$$

where J is the Jacobian matrix of function error derivatives, J' is the transpose of J, error e is the difference between desired and calculated output, I is the identity matrix, and is a scalar quantity that is well defined in. 70 percent of the total database is used for training at random. Finally, testing is carried out on the remaining 30% of data. By updating the weight of the hidden and output layers of the NN, the error between expected and known output was

reduced. As the number of epochs in the neural network grows, the Mean Square Error (MSE) for training and testing drops to an order of 10^{-2} from 10^{0}. The neural network achieved a minimal MSE of for broadband absorber and for multiband absorber after 100 iterations.

After training and testing the neural network, we looked for only those parameters for which the mean square error was the smallest, indicating that those parameters were the best optimised by the neural network model.

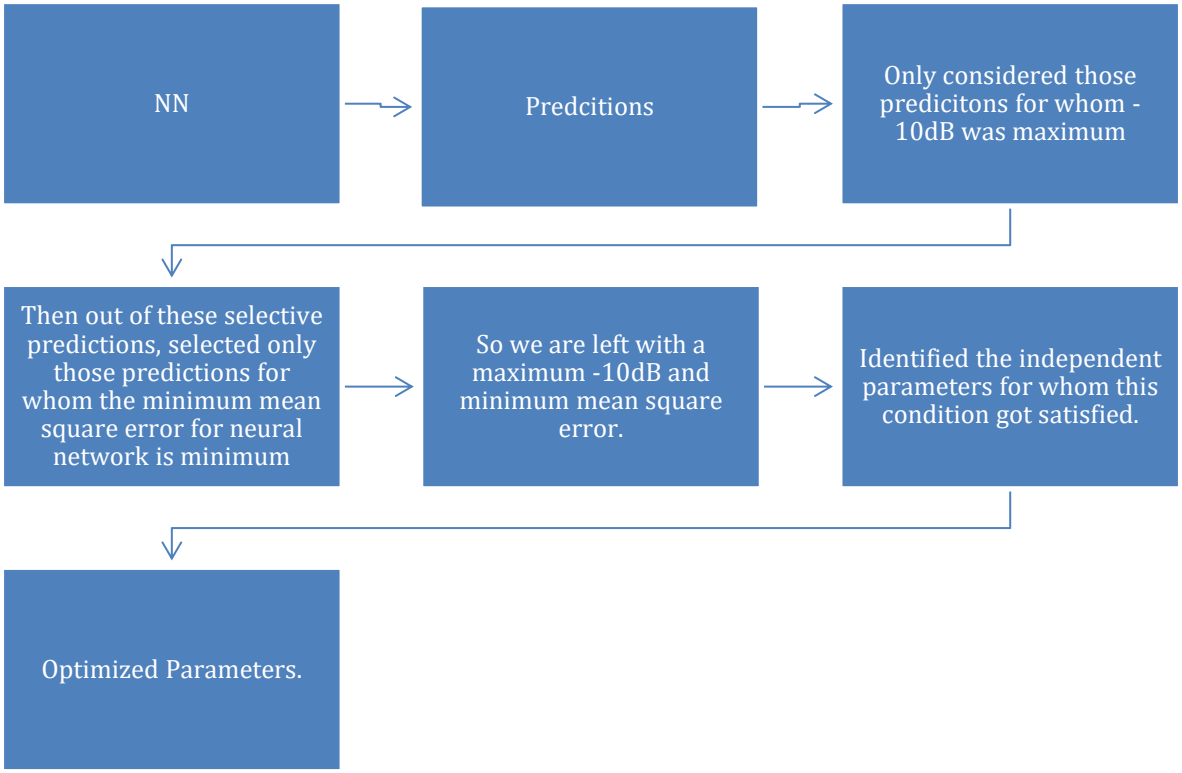


Figure 6.1 : Representing the graphical view about how optimized parameters have been choosen

The results of neural network trained by LM algorithm were as follows :-

Absorber	a	b	l1	w1	l2	w2	l3	w3	t	-10dB	-10dB(predicted)	Min mse
Boradband	9.8	9.05	7.5	1.9	7.6	3	7.5	2.8	0.2	0.56	0.5381	0.0004774
Multiband	9.1	9.1	7.3	2.8	7.7	2.45	7.5	3	0.2	0.49	0.4903	0.015214603

Table 6.1 : Result from neural network

- It is clearly visible that our neural network is able to make very close predictions.

Performance Metric for our neural network :-

The performance metric for our neural network is Mean Square Error. The error E, also known as the cost function, can be determined in a variety of ways when it comes to neural network training. The mean squared error is used in every computation in Pyrenn, which is required when using the Levenberg-Marquardt approach.

Mean Square Error :- The mean squared error (MSE) is calculated by squareing the difference between your model's predictions and the actual results, then averaging that value across the whole dataset.The MSE will never be negative because we square all of our errors. The MSE is calculated using the equation below.:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Fig :- Shows the mathematical expression of MSE

Where N represents the total number of samples that will be evaluated.

Advantages of MSE :-

Because the MSE gives these errors more weight because of the squaring portion of the function, it's a great tool for ensuring that our trained model doesn't have any outlier predictions with large errors.

Disadvantages of MSE :-

The squaring function amplifies the error if our model produces a single inaccurate prediction. In many circumstances, though, we don't worry about the outliers and instead aim for a well-rounded model that performs well enough on the majority of the data.

CHAPTER 07: Results

For Broadband Absorber :-

The dimensions of the Broadband absorber structure is listed in Table given below.

Absorber	a(mm)	b(mm)	l1(mm)	w1(mm)	l2(mm)	w2(mm)	l3(mm)	w3(mm)	t(mm)
Broadband	9.8	9.05	7.5	1.9	7.6	3	7.5	2.8	0.2

Table 7.1 : Optimized parameters for broadband absorber

Reflection coefficient - Broadband absorber (in fig 7.1) having three distinct reflection dips at 9.29, 9.48, and 9.69 GHz, respectively, with reflection minima of -15.6113, -16.6720, and -15.1903 dB**.**

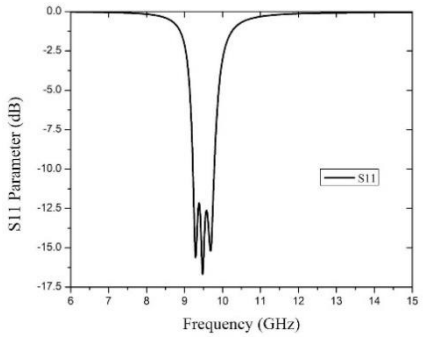


Figure 7.1 : Broadband Reflection Coefficient

Absorption - Broadband with the parameters(in fig 7.2) listed in the table above and a centre frequency of 10.5 has a maximum absorption of 85 percent. As a result, this absorber can be employed in applications requiring up to 85 percent absorption.

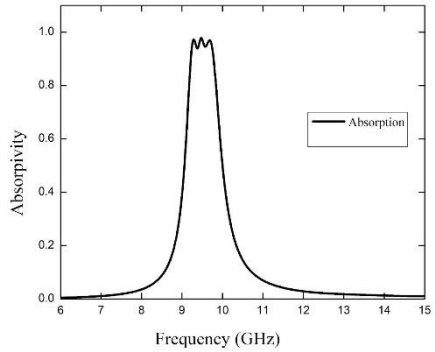


Figure 7.2 : Broadband Absorption

Because of its fourfold symmetry, the proposed Broadband absorber structure is polarisation insensitive. As illustrated in Figures(fig 7.3,7.4) , the 10 dB absorber was also tested for oblique incidence under both TE and TM polarizations. The structure maintains bandwidth-enhanced absorption up to a 45° incidence angle, as shown in these figures. The polarisation of the incident light was used to discover this.

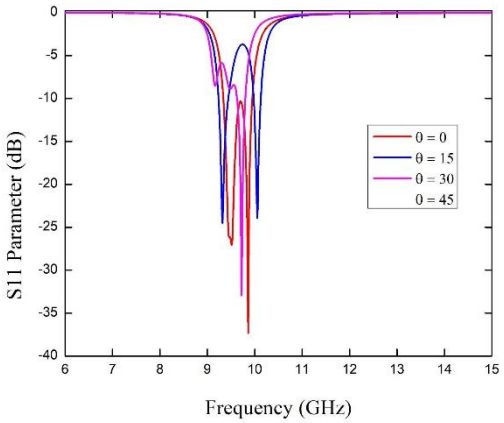


Figure 7.3 : Broadband Theta

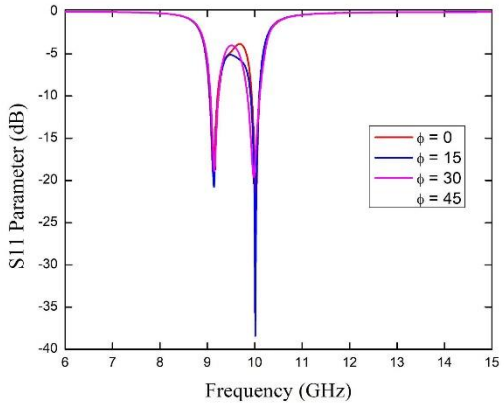


Figure 7.4 : Broadband Phi

Multiband Absorber :-

Multiband :- The simulated reflection coefficient spectra for a Multiband absorber are shown in this diagram. The Multiband absorber structure's dimensions are stated in the table below.

Absorber	a	b	l1	w1	l2	w2	l3	w2	t
Multiband	9.1	9.1	7.3	2.8	7.7	2.45	7.5	3	0.2

Table 7.2: Optimized parameters for multiband absorber

Reflection coefficient - Multiband absorber with three different reflection dips(in figure 7.5) at 9.21, 9.48, and 10 GHz with reflection minima of -19.124, -13.5444, and -15.6793 dB, respectively.

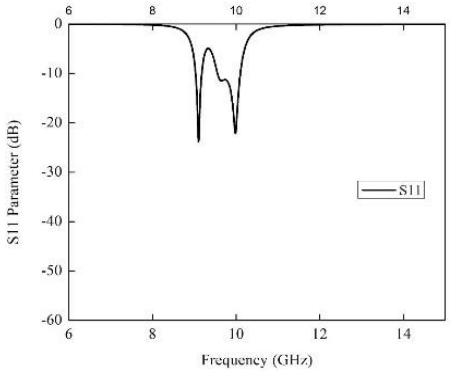


Figure7.5: Multiband Reflection Coefficient

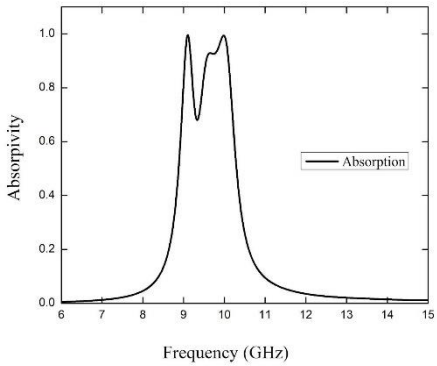


Figure7.6: Multiband Absorption

Absorption - For Multiband with Parameters given in the above table with central frequency 10.5 maximum absorption is at 62%(in fig 7.6). And thus this absorber can be used for applications where till 62% absorption is required. Here the broadband absorber can also be used as it has 85% absorption.

The proposed structure for a broadband absorber is immune to polarisation since it possesses fourfold symmetry. This makes it an ideal candidate for the design. As can be seen in Fig 7.7,7.8 respectively, the 10 dB absorber was further investigated in terms of its oblique incidence when subjected to TE and TM polarizations. It is clear from looking at these numbers that the structure keeps its bandwidth-enhanced absorption even when the incidence angle is 45°. Examining the polarisation of the light that was coming in through the window revealed this fact. As it was observed in Broadband Absorber Multiband also exhibits the same characteristics as it should be.

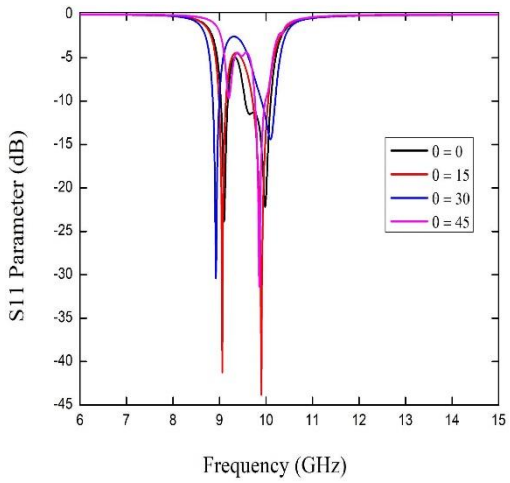


Figure7.7: Multiband Phi

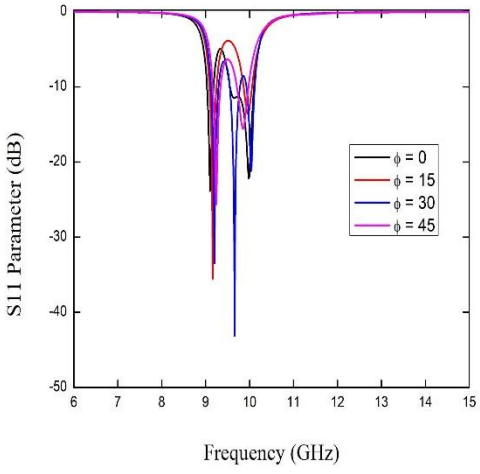


Figure7.8: Multiband Theta

Mentioning of J/H intensity distribution

The bottom layer provides the lowest absorption frequency (9.29GHz)(in fig 7.10), middle layer generates the intermediate frequency (9.48 GHz)(in fig 7.9), and the top layer exhibits the highest absorption frequency (9.69 GHz)(in fig 7.11).

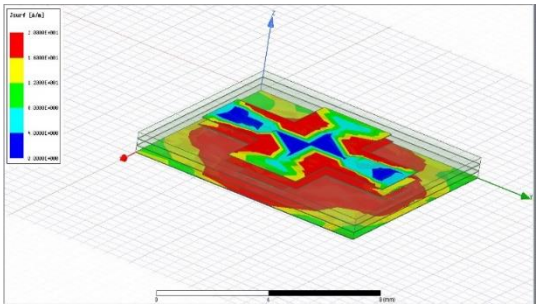


Figure 7.9: Surface current middle layer

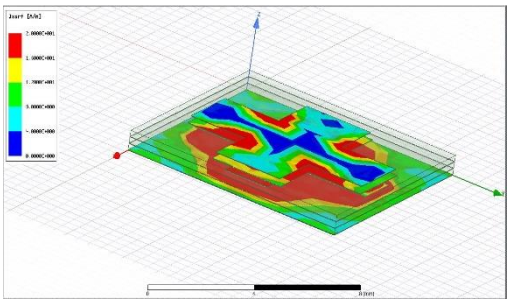


Figure7.10: Surface current Bottom Layer

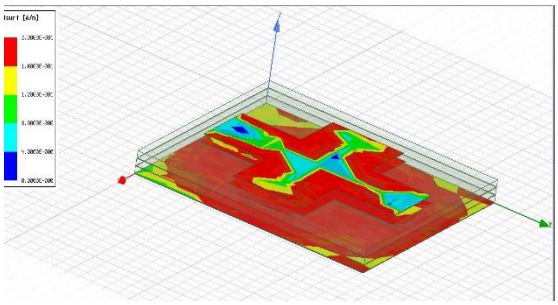


Figure7.11: Surface current Top layer

Chapter 08: Conclusion and Future Scope

Final Conclusion

In this thesis work, a hfss based polarization-insensitive, angular stable GHz microwave absorber is introduced by using a fast and more accurate neural network approach validated by the results from ansys simulation software. For broadband absorber the peak absorption was observed at 62% at set of input parameters which are mentioned in table 10, and for the multiband absorber the peak absorption was observed at 85% at set of input parameters which are discussed in table 10. Furthermore the discussed absorber works great with both TE and TM polarizations up to 45^0.

The best parameters for Broadband and Multiband absorbers are as follows :-

Absorber	a(*mm*)	b*(mm)*	L1*(mm)*	W1*(mm)*	L2*(mm)*	W2*(mm)*	L3*(mm)*	W3*(mm)*	T*(mm)*
Broadband	9.8	9.05	7.5	1.9	7.6	3	7.5	2.8	0.2
Multiband	9.1	9.1	7.3	2.8	7.7	2.45	7.5	3	0.2

Table 8.1: Optimized parameters

Future Scope

In terms of future work, the best future work that can be carried on with this problem statement is to create a neural network architecture from scratch. It would obviously won’t be easy and would be tedious job but at the end it will give you more control over optimizing the algorithm for getting the best results.

References

1. R. Panwar and J. R. Lee, “Recent advances in thin and broadband layered microwave absorbing and shielding structures for commercial and defense applications,” Funct. Compos. Struct., vol. 1, no. 3, Jul. 2019,Art. no. 032001.
2. G. Varshney, “Wideband THz absorber: By merging the resonance of dielectric cavity and graphite disk resonator,” IEEE Sensors J., vol. 21, no. 2, pp. 1635–1643, Jan. 2021.
3. Y. Wen, W. Ma, J. Bailey, G. Matmon, and X. Yu, “Broadband terahertz metamaterial absorber based on asymmetric resonators with perfect absorption,” IEEE Trans. THz Sci. Technol., vol. 5, no. 3, pp. 406–411, May 2015.
4. J. M. Woo, M.-S. Kim, H. W. Kim, and J.-H. Jang, “Graphene based salisbury screen for terahertz absorber,” Appl. Phys. Lett., vol. 104, no. 8, 2014, Art. no. 081106.
5. R. Panwar, S. Puthucheri, V. Agarwala, and D. Singh, “Fractal frequency-selective surface embedded thin broadband microwave absorber coatings using heterogeneous composites,” IEEE Trans. Microw. Theory Techn., vol. 63, no. 8, pp. 2438–2448, Aug. 2015.
6. S. Asgari, N. Sharifi, and N. Granpayeh, “Active tunable terahertz microelectromechanical metamaterial absorber,” J. Micromech. Microeng., vol. 29, no. 4, Apr. 2019, Art. no. 045010.
7. H. Li, J. Niu, and G. Wang, “Dual-band, polarization-insensitive metamaterial perfect absorber based on monolayer graphene in the midinfrared range,” Results Phys., vol. 13, Jun. 2019, Art. no. 102313.
8. S. Asgari and T. Fabritius, “Equivalent circuit model of graphene chiral multi-band metadevice absorber composed of U-shaped resonator array,” Opt. Exp., vol. 28, no. 26, pp. 39850–39867, 2020.
9. H. Zou and Y. Cheng, “Design of a six-band terahertz metamaterial absorber for temperature sensing application,” Opt. Mater., vol. 88, pp. 674–679, Feb. 2019.
10. V. Chaudhary and R. Panwar, “FSS derived using a new equivalent circuit model backed deep neural network,” IEEE Antennas Wireless Propag. Lett., early access, Aug. 2, 2021, doi: 10.1109/LAWP.2021.3101225.
11. Varun Chaudhary, Ravi Panwar. "Neural Network Topology Based Terahertz Absorber Using Fractal Frequency Selective Surface" , IEEE Sensors Journal, 2021