A Project Report

on

Microstrip Line Fed Dielectric Resonator Antenna Optimization using ML Algorithms

Submitted in the Partial Fulfillment of the Requirements
for the award of

Bachelor of Technology
in
Electronics & Communication Engineering

By

Om Singh
Prateek
Pritam Rauniyar

Under the guidance of

Dr. Anand Sharma
Assistant Professor

Department of Electronics & Communication Engineering
Motilal Nehru National Institute of Technology Allahabad

Table of Content

Certificate Pg No. I

Similarity Index Certificate from IPR Cell Pg No. II

Acknowledgment Pg No. III

Abstract Pg No. IV

List of figures Pg No. V

Abbreviations Pg No. VI

Chapter 1: Outline

1.1 Introduction Pg No. 1
1.2 Objective Pg No. 1

Chapter 2: Literature Review

2.1 Cylindrical DRA Pg No. 2
2.2 Machine Learning Pg No. 3
2.2.1 ANN Pg No. 4
2.2.2 Decision Tree Pg No. 5
2.2.3 Random Forest Pg No. 5
2.2.4 KNN Pg No. 6
2.2.5 XGBoost Pg No. 7

Chapter 3: Implementation

3.1 HFSS Model Pg No. 8
3.2 Dataset Pg No. 9
3.3 Training Pg No. 10

Chapter 4: Analysis

4.1 Prediction Pg No. 15
4.2 Accuracy Pg No. 16
4.3 Comparison Pg No. 17

Chapter 5: Improving ANN

5.1 Unexpected ANN results Pg No. 20
5.2 KBNN Pg No. 20
5.3 Implementation Pg No. 21
5.4 Training Pg No. 21

Chapter 6: Fabrication

6.1 Antenna Geometry Pg No. 24
6.2 Outcomes Pg No. 25

Conclusion Pg No. 27
Future Scope Pg No. 28
References Pg No. 29

Certificate

This is to certify that the work contained in the thesis titled **Antenna Design with Machine Learning**, submitted by **Om Singh**, **Prateek** and **Pritam Rauniyar** in the partial fulfillment of the requirement for the award of Bachelor of Technology in Electronics and Communication Engineering to the Electronics and Communication Engineering Department, Motilal Nehru National Institute of Technology, Allahabad, is a bonafide work of the students carried out under my supervision.

Date: 13th May 2022

Place: Prayagraj, India

Dr. Anand Sharma
Assistant Professor
ECE Department

MNNIT, Allahabad

Similarity Index Certificate from IPR Cell

Acknowledgment

Foremost, I would like to express my sincere gratitude to our project supervisor Dr. Anand Sharma Sir, for his guidance and mentorship. Our team faced several roadblocks during the project development; sir helped us get through that and be clear with the concepts and perspective. Our team is glad to have a supervisor who provided us with such quality concept clarity and was always available for our curious questions.

We would like to thank Dr. P. Ranjan (Assistant Professor: IITM Electrical /Electronics Dept), who made the project roadmap clear in the initial phase and helped us visualize subsequent major tasks. Following his timeline, we were able to get done with most of our study and research work in a timely and regular fashion. The weekly and bi-weekly scheduled calls proved crucial in being on the right track, making our research fruitful and clear.

Our sincere thanks also go to Harshit Gupta Sir, whose guidance helped us perform the implementation, producing better outcomes and results of each analysis. He also taught us to be to the point while presenting our study in different evaluation rounds and how to make presentations concise. He also guided us with the results to give a descriptive picture of our research and implementation.

Abstract

In this report, our team is attempting to implement various machine learning techniques into Antenna Design Optimization. As a reference antenna, our team has used a Cylindrical DRA (Di-electric Resonator Antenna) with appropriate design parameters. This Antenna Model is designed in the HFSS(High-Frequency Structure Simulator) environment, using which we generated the data-set and verified results.

Studying these ML algorithms, we are trying to prove the efficiency and reliability of these techniques over conventional optimization methods. Each method is analyzed adequately by first training the learning model with the generated data-set. Training is followed by predictions made by each model for given input, comparing the prediction with actual results helped in the accuracy analysis. Further a detailed comparative analysis of these predicted values with the HFSS results was carried out to verify the accuracy. Alongside implementing conventional ANN, some work on some Knowledge-Based Neural Network techniques has also been done, followed by a comparative analysis.

With our study we tried to conclude the best possible algorithms and also presented the future prospects of work which can be carried on further in the same direction. All these optimization techniques were also compared to opt out the most suitable algorithm for specific use-cases.

List of figures

Fig 2.1: Antenna configuration of cylindrical DRA	Pg No. 2
Fig 2.2: The typical Artificial Neural Network	Pg No. 4
Fig 2.3: ANN Architecture based on MLP	Pg No. 4
Fig 2.4**:** Decision Tree structure	Pg No.5
Fig 2.5**:** Categorizing a new data point by KNN	Pg No.6
Fig 3.1**:** Designed HFSS model of CDRA	Pg No.7
Fig 3.2**:** Optimization iterations cost values.	Pg No. 8
Fig 3.3: S11 sweep for each iteration	Pg No. 8
Fig 3.4**:** Loss vs Epoch graph for ANN	Pg No. 11
Fig 3.5**:** ANN Model validation loss vs Epoch graph	Pg No. 11
Fig 4.1 Predicted vs Actual values for ANN	Pg No. 14
Fig 4.2 Predicted vs Actual values for Decision Tree	Pg No. 14
Fig 4.3 Predicted vs Actual values for KNN	Pg No. 15
Fig 4.4 Predicted vs Actual values for Random Forest	Pg No. 15
Fig 4.5 Predicted vs Actual values for XGBoost	Pg No. 15
Fig 5.1 Error matrix is generated on which the ANN is trained	Pg No. 18
Fig 5.2 Predicting the FOM for particular input parameters	Pg No. 18
Fig 5.3 Prior Knowledge Input KBNN	Pg No. 18
Fig 5.4 PKI accuracy Chart	Pg No. 18
Fig 5.5 Source Difference Accuracy Chart	Pg No. 18
Fig 6.1 Microstrip Line fed cylindrical DRA	Pg No. 24
Fig 6.2 Pictures of Fabricated Antenna	Pg No. 24
Fig 6.3 Measured/Simulated $\|S_{11}\|$ of the proposed antenna	Pg No. 25
Fig 6.4 E-field variation of proposed antenna at 3.5 GHz	Pg No. 26
Fig 6.5 Measured/Simulated Gain variation of proposed antenna	Pg No. 26
Fig 6.6 Far-field pattern of proposed antenna at 3.5 GHz	Pg No. 27

List of abbreviations

1. CDRA Cylindrical Dielectric Resonator Antenna
2. ANN Artificial Neural Network
3. KNN K Nearest Neighbors
4. ML Machine Learning
5. DL Deep Learning
6. HFSS High Frequency Structure Simulator
7. EM Electro-Magnetic
8. MLP Multilayer-perceptron
9. LM Levenberg-Marquardt
10. FOM Figure of Merit
11. IoT Internet of Things

Chapter 1

Outline

1.1 Introduction

Due to advancements in various technical sectors, the requirement for varied antenna design has increased exponentially. Considering IoT, for instance, different device dimensions demand antennas with precise parameters; thus, these antennas must be fabricated with high precision keeping in account an optimized performance. These days optimization is carried out mostly with traditional methods (primarily using EM simulators) which have critical down points like high compute intensity, less accurate results. To optimize an antenna having several parameters which have to be optimized (e.g., 3D Antennas) [1] these traditional methods are proved impractical. To tackle these design challenges Machine Learning Domain is getting considered a suitable tool for this use case. In this communication we are trying to do a thorough study of a CDRA (Cylindrical Dielectric Resonator Antenna) to obtain the most optimal Machine Learning technique applicable, and the study can then be extended to another antenna designs also.

1.2 Objective

The Antenna Design optimization process can leverage some of the positives offered by Machine Learning techniques, this could tackle several issues related to design optimization. The study throws some light on this particular use case of ML with Antennas and will try to put forward different methodologies through comparison. We will identify the effectiveness and accuracy of various Machine Learning Techniques on our antenna and once we have obtained the techniques and have the relational model, the output for any data model can be predicted while using the same data set for multiple goals.

Chapter 2

Literature Review

2.1 CDRA

CDRA stands for Cylindrical Dielectric Resonator Antenna which as evident from the figure, cylindrical attributes to its shape. It is utilized more often than not at frequencies greater than or equal to microwave frequencies (> 1 GHz) and thus acts like a radio antenna. The Dielectric Resonator is made up of ceramic, which is the dielectric here, and is placed on a metal surface, ground plane. The choice of ceramic is due to the requirement of a material with high quality factor ($20 < Q < 1000$) and having low loss property. Ideally, we require the dielectric constant between $10 < \varepsilon r < 100$ so that we can trade off with other factors for various applications. DRA's can perform the transformation of guided waves into unguided waves (RF signals) using the radiating resonators. The size of DRA is inversely proportional to $(\varepsilon r)^{\wedge}½$ and hence can be contained in a small space if the chosen material is of high dielectric constant. Courtesy of this, DRA's are now looked upon to be the possible solution to the requirement of smaller spacecraft equipment in future. Also, the choice of low-loss material and the absence of any conducting material results in excellent radiation efficiency, thus making it optimal for applications operating at high frequencies. The resonant frequency of a CDRA can be roughly represented by, the resonant frequency is the frequency corresponding to the minima in the S11 sweep of the antenna.

$$f_r = \frac{c}{2\pi r\sqrt{\varepsilon DRA}}\left[1.71 + \frac{r}{h} + 0.1578\left(\frac{r}{h}\right)^2\right]$$

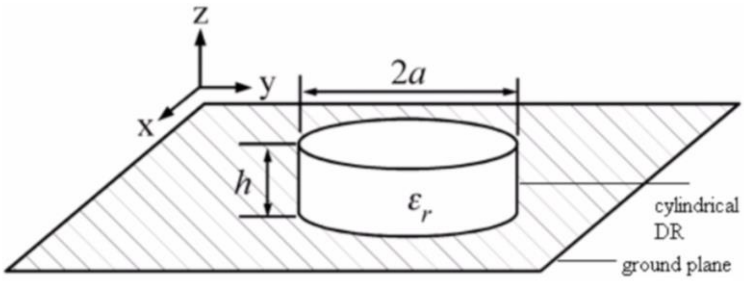


Fig 2.1: Antenna configuration of cylindrical DRA

2.2 Machine Learning

A domain that recently has proved very useful with decision making and analytics in a wide variety of applications. With techniques and optimization algorithms we can find the hidden mathematical relations in the input and output data considering which we can make future predictions. Other heuristic optimization techniques could also be used like genetic algorithms and particle swarm optimization but these algorithms search for the optimal solution by analyzing the output on individual data points and generating new and possibly better search directions until one global maxima or minima is identified. And once a model is trained the same data set gets beneficial for multiple output goals. Though some studies regarding ML implementation have been conducted as mentioned in [2], a comparative study could be required to get a clear picture. Our work tries to support ML as a promising choice to include in antenna design, and later on, this same idea can be extended to complicated designs even. Below mentioned 5 ML techniques have been studied, implemented, and analyzed thoroughly.

Broadly classifying there are two types of ML techniques - supervised learning and unsupervised learning. The major difference between the two techniques is the use of labeled data to train. Supervised techniques use labeled data and the model has the job of finding the mapping function. Whereas unsupervised learning uses unlabeled data and the model here itself has to infer patterns from the data set. Unsupervised learning models find the similarities and kind of cluster or group them together. It can

be used to gather important insights from the dataset. However, since here we are trying to build a model which can predict the output for any data provided, we had to go forward with supervised learning models. Unsupervised learning is closer to Artificial Intelligence compared to supervised learning as it learns similarly as a child by observing its surroundings. But supervised learning is optimal for applications requiring an accurate response and therefore we can justify the use of it in our scenario.

2.2.1 ANN (Artificial Neural Network)

The technique comes under Deep Learning, a subset of Machine Learning which is inspired by biological brain functionality. It consists of a group of artificial neurons which process info over intercommunication. Drawing an analogy with our biological neural network - dendrites represent inputs, cell nucleus represents nodes, synapse represents weights and axon represents output. This tries to follow up our brain striving for a similar ability to understand things and carry out decision making.

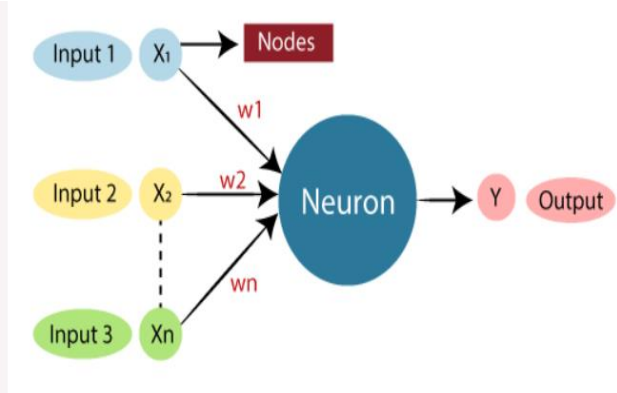


Fig 2.2: The typical Artificial Neural Network

Certain common ANN structures include MLP (Multi-Layer Perceptrons) which consists of 3 layers: Input, Output, and Hidden layer. Training for small and medium-sized patterns can be done with algorithms like LM (Levenberg Marquardt) [3], back-propagation, and delta-bar delta. The prowess to learn and develop complex relationship models gives MLP's an edge over others in the preference order.

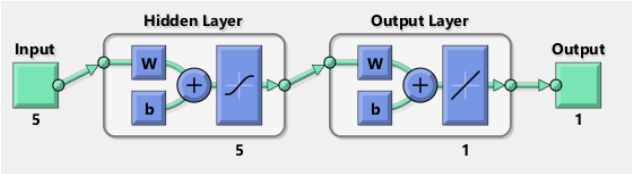


Fig 2.3: ANN Architecture based on MLP

2.2.2 Decision Tree

A decision-making technique that uses a tree-based model, the node, branch, and leaves respectively represent attribute test, outcome, and class label. It is generally represented as a flow chart and is a popular tool for classification and prediction applications. Decision Trees can be classified on the basis of type of target variable - Categorical or Continuous. The nodes here also allow splitting into two or more sub nodes. In the decision tree each node is a classifier and decides which edge the entity will descend to based on the attribute of the node. This is a recursive process and is performed all the way from the root node till the terminal node - which is the final classification of the entity.

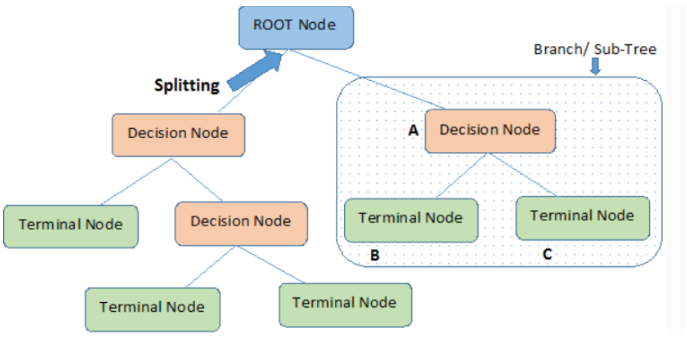


Fig 2.4: Decision Tree structure

However, this model can have prediction drawbacks in case of high noise in the training data. Moreover, a small change in the training data could change the structure of the optimal tree which can result in inaccuracy.

2.2.3 Random Forest

Based on the decision tree Random Forest algorithm is widely used for classification and regression. It basically consists of a number of decision trees made

on different data subsets; an average is taken. It's an extension of the Bagging (bootstrapping + aggregation) technique and is based on the concept of ensemble learning.

2.2.4 KNN (K-Nearest Neighbor) [4]

A supervised and insurance-based machine learning technique that makes predictions considering the similarities with the training data set. It takes a new data point and then using its algo places it into the category which has the most similarity out of all available categories. It is not an immediate learner from the training sets, rather it conserves all the data and takes action at the time of classification. There is no definite way for the determination of K - it can be perfected using hit and trial only. After selecting K, we need to figure out the euclidean distance of its neighbors and proceed to picking out the K closest ones to the new data point. Once we have these K neighbors, make a count of the number of data points falling in each category. The category with the maximum number of closest data points will have the new data point assigned to it.

Fig 2.5: Categorizing a new data point by KNN

The value of k is considered 5 in our antenna's case as it gives the minimum cross-validation error.

2.2.5 XGBoost (Extreme Gradient Boost) [5]

Decision tree-based algorithms work on ensemble learning and gradient boosting. It is an optimized distributed gradient boosting library that is highly efficient, flexible, portable, and provides a parallel tree boosting that solves many data science problems in a fast and accurate way. It is widely used as an alternative to ANN in applications involving small/medium-sized structured/tabular data.

Chapter 3

Implementation

3.1 HFSS Model

High-Frequency Structure Simulator, is an industry-standard software tool distributed by Ansys that is used for antenna design, complex radio frequency electronic circuit elements including transmission lines, filters, etc. [6] The reference cylindrical DRA is designed and tested in an HFSS environment. The performance of a CDRA depends on several design parameters, for our study we have considered the *h* and *r* (height and the radius) of the cylindrical substrate as variable parameters. The frequency band is 1-5 GHz, and we have considered the Figure of merit as S11 value generated with variable *h, r,* and *f* values (*f* = operation frequency). S11 is the reflection coefficient which determines the power which is reflected from the antenna. This parameter could determine the antennas performance in major cases and thus has been selected as an FOM.

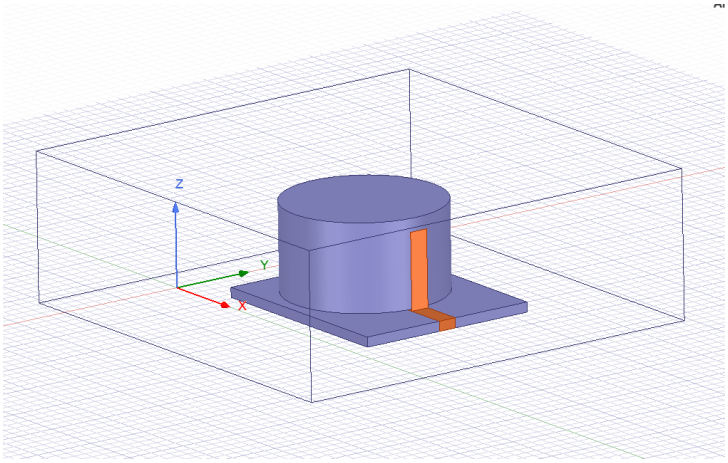


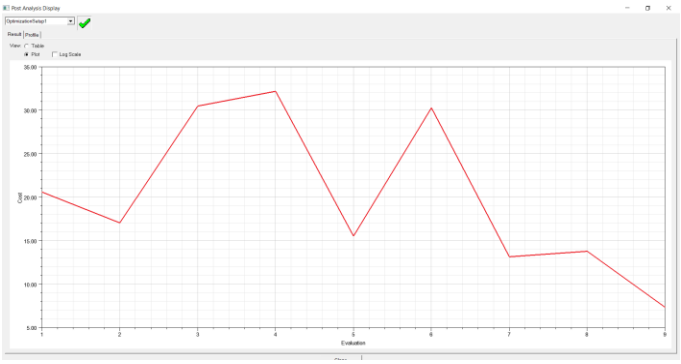
Fig 3.1: Designed HFSS model of CDRA

For the implementation purpose, firstly we have designed this CDRA on Ansys HFSS. Few design parameters which have been considered are (substrate), r

(substrate radius), w1 (width of the feedline), w2 (width of the vertical feed line). These parameters are chosen as they significantly affect the S11 which is the FOM in most design requirements. Scattering parameters define the properties of microwave devices, S11 in this case refers to the amount of power that is reflected back when the antenna is fed with power, and its value should be as low as possible. However not only does S11 get to be the FOM in each requirement, but there could also be other constraints that could introduce tradeoffs. For our study, we will consider only S11 as the desired FOM. Working with a conventional EM simulator gets highly inefficient, as each design iteration requires high compute which makes the entire process time-consuming. Even though each operation needs proper manual supervision and expertise, automation in this entire process is a great requirement. To generate the dataset from HFSS we have implemented HFSS parametric.

3.2 Dataset

For the training purpose, the model requires proper data to get an accurate result. On increasing the training samples the model prediction gets better. For the dataset generation, HFSS Parametric is used. But we need a range and step size for *h, r, w1, w2* first, and for this firstly, we will try to optimize the design with HFSS optimitrics [7]. Here our optimization goal is to get the dB(S11) value <= -20 dB in the frequency band of 1-5Ghz. There are options for several optimizers, but for this particular use case, we have used the Quasi-Newton *(Gradient)* optimizer. On optimization with 9 iterations, we got the optimized value of *h, r, w1 and w2.*



The graph shows the values of cost for all 9 iterations. The parameter values of the iteration with the least value of cost are considered optimal.

Fig 3.2: Optimization iterations cost values.

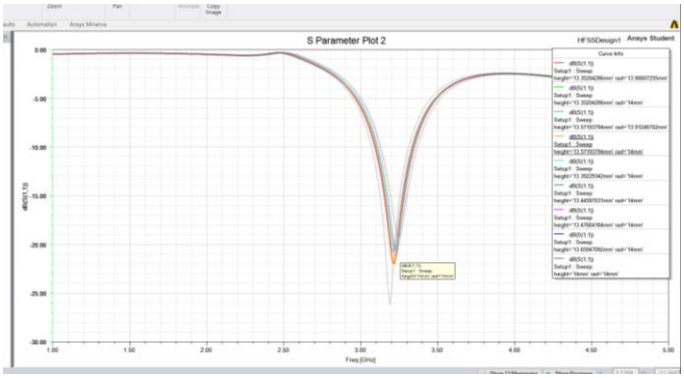


Fig 3.3: S11 sweep for each iteration, *9 S11 sweep graphs are overlapped representing the dB S11 values for different frequencies*.

With the optimized values, we can proceed with HFSS parametric. By setting a suitable range and step size of *h , r, w1 and w2* we can generate a dataset that can be exported in any suitable format. We have considered the .csv format as the data can be visualized on any spreadsheet and can be easily parsed.

The range and step size selected for h,r, w1 and w2 are 13mm - 14mm: 0.2mm, 13mm-14.2mm: 0.4mm, 2mm - 3mm: 0.5mm, 2mm - 3mm: 0.5 mm. These ranges result in 216 parameter combinations, which means HFSS will have to perform 216 parameter operations. This process of generating the dataset with such minor iteration count itself shows the inefficiency of using such simulation software for parameter explorations or for complex antenna design optimization.

3.3 Training

The generated dataset in .csv format is first parsed and input/output data frames are made with it. Data-frames are then distributed into training and test sets. The training set will be used for training each model, which will then be making predictions for test sets. The accuracy of the test-set predictions can then be analyzed to determine the feasibility of the model. This distribution will be random with test-data size being 30% of the total set.

```
# Reading data from csv file

dataset = pd.read_csv('../DataSets/data_set.csv')

x = pd.DataFrame(dataset.iloc[:, 0:3].values)
y = dataset.iloc[:, 3].values

# Generating training and test data sets

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 123)
```

The python code of each model's definition and training is mentioned in the following section.

KNN

```
# KNN

from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(x_train, y_train)
```

Used the KNeighborsRegressor class from the *sklearn* library, k= 5 is set for the regressor.

ANN

```
# Artificial NN

classifier = Sequential()

# Input Layer
classifier.add(Dense(units = 6, activation = 'relu', input_dim = 3))

# 1st Hidden Layer
classifier.add(Dense(units = 8, activation = 'relu'))

# 2nd Hidden Layer
classifier.add(Dense(units = 3, activation = 'relu'))

# Output Layer
classifier.add(Dense(units = 1, activation = 'linear'))
```

Implementation of ANN was done using *keras* library, which internally uses tensor-flow library. There is one input layer with the dimension of 5 as we have five input features for the model with a *'relu'* activation function. 2 hidden layers are added with the same *'relu'* activation function. The output layer of unit=1 provides the final

result along with this the Adam optimizer and *mean_squared_error* was used as the loss function for the model. Adaptive Moment Estimation is a technique for optimizing gradient descent algorithms. When working with huge problems with a lot of data or parameters, the method is quite efficient. It is efficient and takes minimal memory. It's essentially a hybrid of the 'gradient descent with momentum' and the 'RMSP' algorithms.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\left[\frac{\delta L}{\delta w_t}\right]$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\left[\frac{\delta L}{\delta w_t}\right]^2$$

mt = aggregate of gradients at time t [current] (initially, mt = 0)

mt-1 = aggregate of gradients at time t-1 [previous]

∂L = derivative of Loss Function

∂Wt = derivative of weights at time t

β1 & β2 = decay rates of average of gradients in the above two methods.

```
# Training

classifier.compile(optimizer = 'adam', loss = 'mean_squared_error', metrics = ['accuracy'])

history=classifier.fit(x_train, y_train, batch_size = 10, epochs = 100, validation_data=(x_test,y_test))
```

Used the *Adam* optimizer and *mean_squared_error* as the loss for the model. The training is carried on for 100 epochs with a batch size of 10. Training of an ANN model is the time-taking process for machines with less compute ability.

The training process was also analyzed to have a clear picture of how the model trains itself with the input training data set.

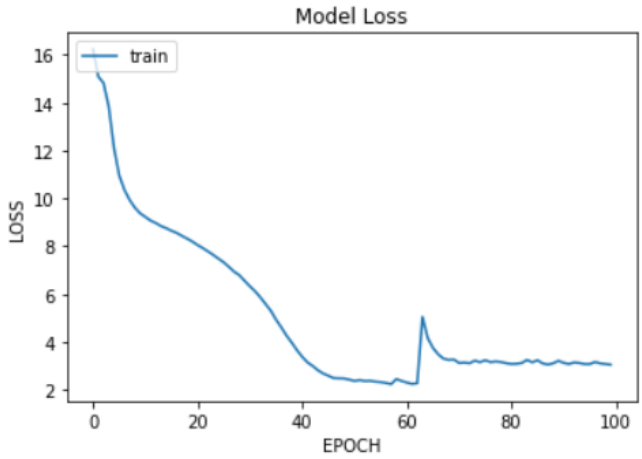


Fig 3.4: Loss vs Epoch graph for ANN

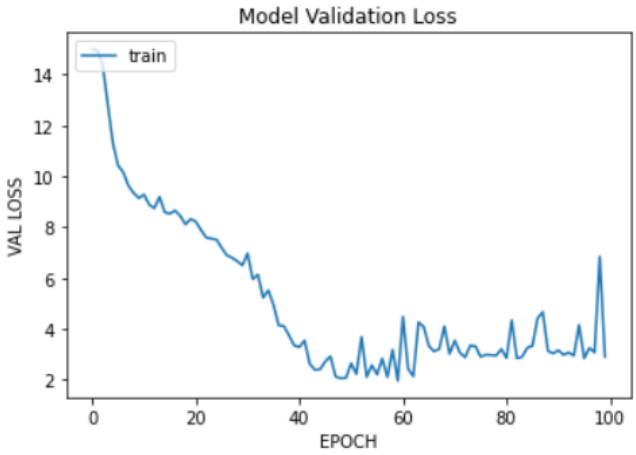


Fig 3.5: ANN Model validation loss vs Epoch graph

Decision Tree

```
# Decision Tree

from sklearn.tree import DecisionTreeRegressor
DTReg = DecisionTreeRegressor()
DTReg.fit(x_train,y_train)
```

Used *sklearn* DecisionTreeRegressor class.

Random Forest

```
# Random Forest

from sklearn.ensemble import RandomForestRegressor
RFreg = RandomForestRegressor(n_estimators = 10, random_state = 0)
RFreg.fit(x_train,y_train)
```

Used *sklearn* RandomForestRegressor class, the value of the estimators is set to 10.

XGBoost

```
# XGBoost
import xgboost as xgb
xgb_reg = xgb.XGBRegressor(n_estimators=100, objective='reg:linear', seed = 123)

# Training
xgb_reg.fit(X_train, y_train)
```

The xgb package is imported and used to create the XGBRegressor.

Machine Learning models for each technique are made on **python notebook** (jupyter), these notebooks are present in the project's **github** repository- https://github.com/OmSingh5092/final-year-project

Chapter 4

Analysis

4.1 Prediction

By now all the models which we are considering are trained with the generated dataset. Each of these models can now predict the values of S11 for a given antenna parameter in a particular frequency band. As we discussed earlier, it is better to test the model with a data-set with which it is not trained, to understand its precision better and analyze the prediction accuracy. If the predictions made are not up to the mark, then any of the following could be the case.

1. The training data could be inappropriate or of poor quality.
2. Model might require hyperparameter tuning or some cross-validation.
3. Choice of the input features could be wrong.

There might be other reasons even, but for our use case, we tried working on the above-mentioned issues in case of the poor model prediction. Prediction metrics provides evaluative and comparative numbers which helps in opting out the best possible algorithm for a specific use case. Two most relied metrics are the r2-score and MSE (mean absolute error)

$$R^2 = 1 - \frac{Sum\ of\ squares\ of\ residuals}{Total\ sum\ of\ squares}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i)^2$$

For instance, in case of ANN if the resultant metrics were not up to the mark the hyperparameters like training epochs, batch size, hidden layers, activation functions were tweaked a bit to have a better performance.

4.2 Accuracy

On comparing the predicted outputs with the actual outputs, we obtained the *r2 score, mean absolute error, mean squared error, max error.* Table 3.1 contains these accuracy metrics for predictions made by each ML technique. To obtain these values we used the methods provided by *sklearn.metrics* . These metrics provide us a picture of the prediction effectiveness and helps in improving the training dataset or tuning the model.

	R2 Score	Mean Absolute Error	Mean Squared Error	Maximum Error
ANN	0.45	1.70	11.04	40.90
KNN	0.98	0.12	0.33	23.53
Decision Tree	0.97	0.18	0.59	22.61
Random Forest	0.99	0.12	0.24	24.21
XGBoost	0.98	0.20	0.37	20.26

Table 4.1: Prediction metrics for each technique

Following are a few graphs that visualize the relations between the predicted and the actual values.

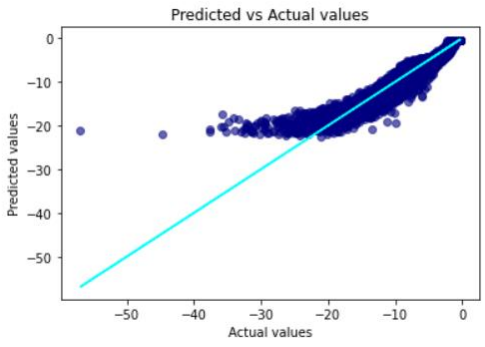


Fig 4.1 Predicted vs Actual values for *ANN*

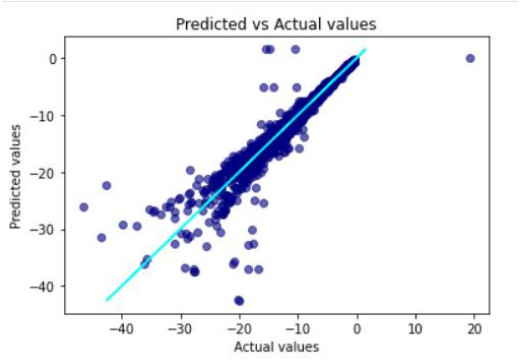


Fig 4.2 Predicted vs Actual values for *Decision Tree*

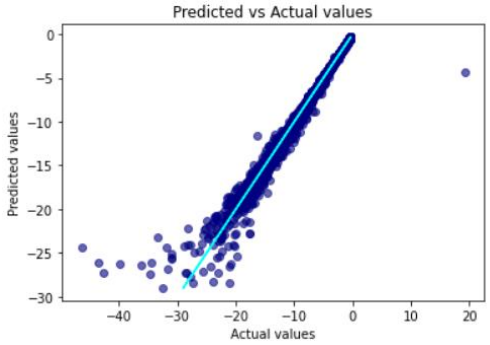


Fig 4.3 Predicted vs Actual values for *KNN*

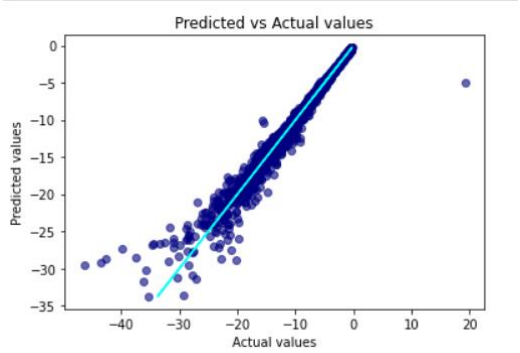


Fig 4.4 Predicted vs Actual values for *Random Forest*

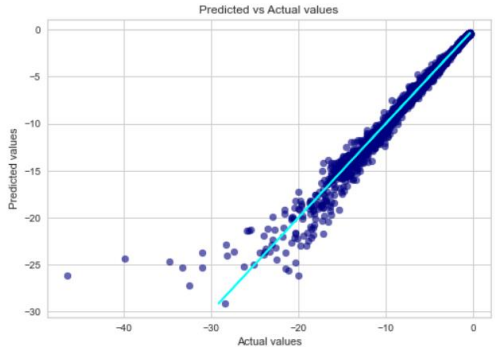


Fig 4.5 Predicted vs Actual values for *XGBoost*

4.3 Comparison

In the accuracy analysis we took a fraction of the data-set (the test-set) to compare the predictions, but now we will carry on the comparison of the HFSS results with the model predicted values. The data which we will use in this section will be randomly generated by uniform random distribution using the *NumPy* library in python. The range and step size values that we used for *h* and *r* in HFSS parametric will only be used here to generate a random data frame with 5 samples. The values of antenna parameters for a given frequency are set for the HFSS model and a simulation is carried out to get the value of S11 which can then be compared with the predicted values. For each data sample, an error is calculated to get an idea of the prediction accuracy considering the HFSS value as the actual value.

Sample	Height (mm)	Radius (mm)	Freq (GHz)	W1 (mm)	W2 (mm)	HFSS S11
D1	14.10	14.55	2.42	2.25	2.6	-0.37
D2	13.04	13.30	1.84	2.3	2.7	-0.43
D3	13.19	13.74	1.95	2.7	2.45	-0.44
D4	14.42	13.12	2.41	2.4	2.8	-0.51
D5	13.86	14.57	2.56	2.8	2.1	-0.77

Table 4.2 Five Random data samples.

	D1	D2	D3	D4	D5
Prediction	-0.31	-0.40	-0.43	-0.64	-0.48
Error	0.06	0.03	0.01	0.13	0.29

Table 4.3 Prediction of KNN

	D1	D2	D3	D4	D5
Prediction	-0.47	-0.47	-0.47	-0.47	-0.87
Error	0.10	0.04	0.03	0.04	0.10

Table 4.4 Prediction of ANN

	D1	D2	D3	D4	D5
Prediction	-0.29	-0.41	-0.42	-0.65	-0.47
Error	0.08	0.02	0.02	0.14	0.30

Table 4.5 Prediction of Decision Tree

	D1	D2	D3	D4	D5
Prediction	-0.27	-0.41	-0.42	-0.67	-0.49
Error	0.10	0.02	0.02	0.16	0.28

Table 4.6 Prediction of Random Forest

	D1	D2	D3	D4	D5
Prediction	-0.43	-0.41	-0.41	-0.65	-0.55
Error	0.06	0.02	0.03	0.14	0.20

Table 4.7 Prediction of XGBoost

Fig 4.6 Graph representing the S11 comparison for each Dataset.

Chapter 5
Improving ANN

5.1 ANN Shortcomings

Accuracy and prediction capability of ANN should be way more than other implemented ML algorithms, however in the accuracy analysis the results are not as expected. Primarily the low accuracy results of ANN are a reason for less training data-set on which the model was trained on. ANN can map complex data relations because of different implied layers, but the data required for training should have large samples, for less training samples other conventional models can easily outperform.

Generating large training-samples through HFSS is not a feasible job, as each iteration requires heavy computation which makes the process highly inefficient. However, when the training data is difficult to obtain, design of experiment (DOE) techniques can also be implied [8], but the data requirement could still be too large. To tackle such issues some existing models which we have implemented could be used to represent additional prior knowledge. These models are already trained on limited data-set thus could just predict outputs for specific inputs, this can be leveraged to enhance the training of ANNs.

5.2 KBNN

Knowledge based neural network uses prior knowledge to design interconnections and weights in the initial network. After that it uses a pure inductive method to adjust the weights and to get the complete design. In order to mitigate the high compute requirement of EM solvers in generating the dataset for ANN, some problem specific additional knowledge can be used. This would accelerate the training of ANNs. Various techniques such as source difference, prior knowledge input, and space-mapping have been developed in order to expedite the training of KBNNs over conventional ANNs. Empirical models have to be developed which could represent this additional knowledge.

5.3 Implementing KBNN

Some work has been done on implementing KBNN on microwave components, which focused on reduction of training data requirement for ANNs [9]. This prior knowledge can be implemented using different methods, like *prior knowledge input* (PKI), *source difference* [10]. In the PKI method the ANN is trained to map the same input-output relation but along with the training data other inputs are also added. Additional data is generated using the empirical models, so for PKI the additional training data is generated using *KNN, Decision-Tree, Random-Forest and XGBoost* models. In the source difference method, the ANN is trained on the error (difference) between the HFSS output and the source-model output. The mapping of this error matrix is simpler thus requiring less training.

5.4 Training

Considering λ as a design parameter vector, for a particular vector the S(λ) is the S11 obtained from HFSS and similarly F(λ) obtained from the empirical model. Further finding out the difference between both results in the error matrix.

$$E(\lambda) = S(\lambda) - F(\lambda)$$

The ANN is now trained on the E(λ), the final required value of S11 can be obtained by summation of the ANN prediction with the empirical model's output for a parameter vector.

$$S11(\lambda) = E(\lambda) + F(\lambda)$$

Training efficiency of KBNN supports high dimensional parametric sweeps and design space exploration, however despite the positives there are several drawbacks even. Since empirical models are representing additional knowledge which is employed in the training of ANN, these models must be accurate enough. For several microwave design applications finding the empirical model beforehand

could be a difficult task e.g., hybrid copper-graphene interconnects [10]. Fig 5.2 and Fig 5.3 represent the accuracy plot of PKI and SourceDifference KBNN models. Table 1 contains the prediction metrics, and provides a comparison of the accuracy performance. Both the KBNN techniques provided better accuracy results as compared to ANN as evident from its MSE and R2 Score. The accuracy results of both the SD and PKI techniques are in close similarity.

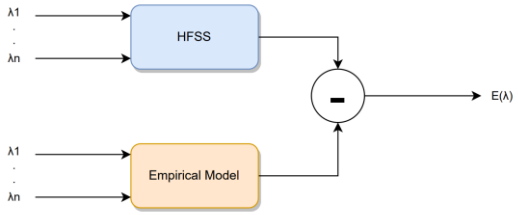


Fig 5.1 Error matrix is generated on which the ANN is trained..

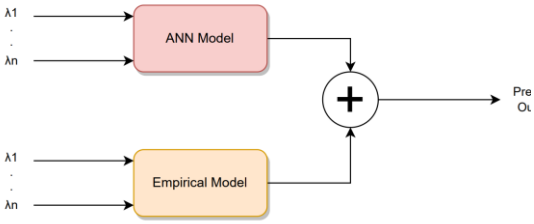


Fig 5.2 Predicting the FOM for particular input parameters.

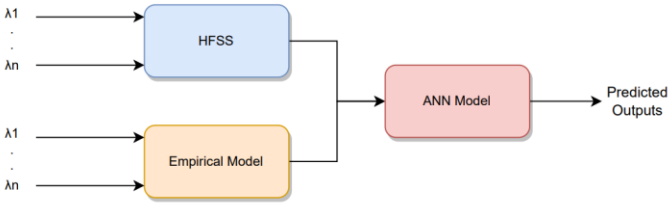


Fig 5.3 Prior Knowledge Input KBNN

	MSE	**R2**
ANN	11.04	0.45
SD KBNN	0.51	0.97
PKI KBNN	0.73	0.96

Table 5.1 Comparison of prediction metrics.

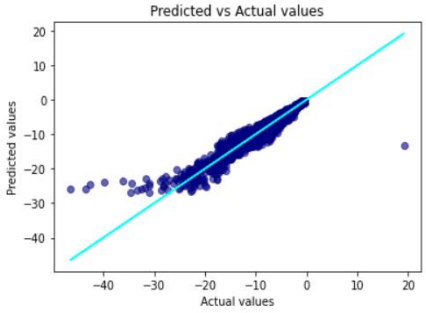


Fig 5.4 PKI accuracy Chart

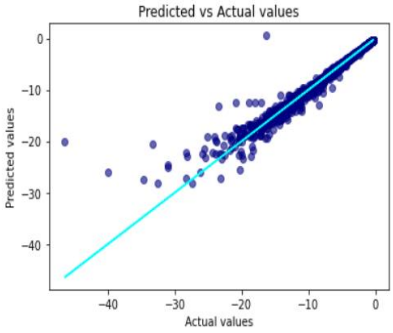


Fig 5.5 Source Difference accuracy chart

Chapter 6

Fabrication

6.1 Antenna Geometry

Fig. 6.1 depicts the planned antenna structure. Alumina-based cylindrical ceramic and a vertical conformal strip are fed through a microstrip line. Cylindrical ceramic has a permittivity of 9.8. The microstrip line was printed on FR4 with a permittivity of 4.4. Fig 6.2 depicts the manufactured antenna's 3D view and feeding structure. The proposed radiator's optimal dimension is presented in the caption of Fig 6.1.

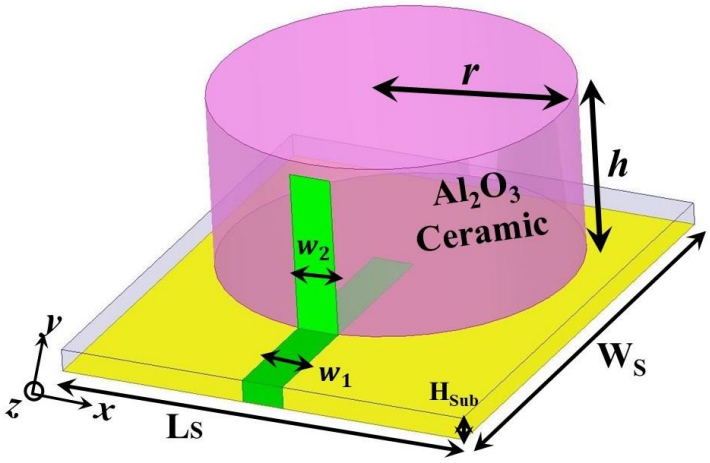
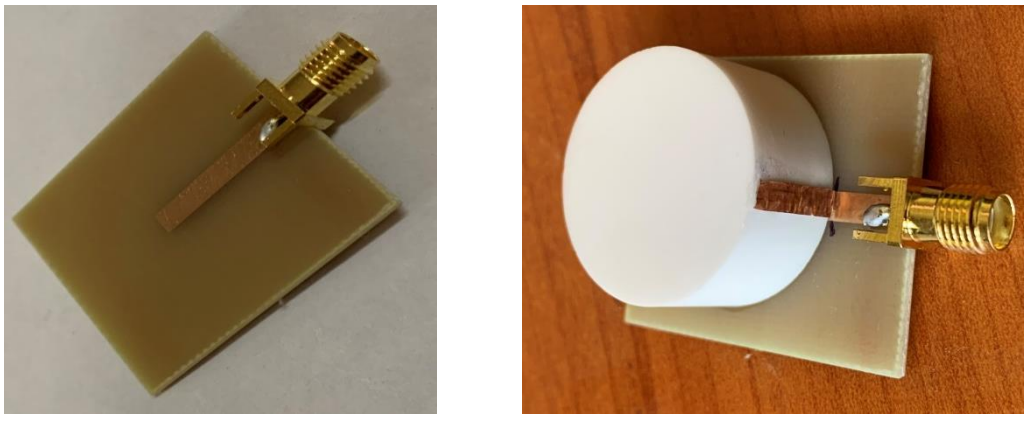


Fig. 6.1 Microstrip Line fed cylindrical DRA; L_S= 30mm; W_S= 35mm; w_1=3.0 mm; w_2=3.0 mm; H_{Sub}=1.6 mm; r=12.5 mm; h=12.5 mm



(a) (b)

Fig. 6.2 Pictures of Fabricated Antenna (a) Feeding Structure (b) 3D View

6.2 Outcomes

The projected outcome is compared to the measured outcomes in this section. The suggested antenna's reflection coefficient (|S11|) is measured using a Keysight VNA E5071C. Figure 6.3 depicts the measured and simulated fluctuation of |S11| in terms of frequency. The good connection between measured and simulated |S11| can easily be seen in Figure 6.3. The proposed antenna works in the 3.3-3.65 GHz frequency band. The E-field pattern is shown in Fig.8 to find the accountability of resonance at 3.5 GHz. HEM11 mode is created in cylindrical-shaped ceramic [11], according to Fig. 8. It's because of the magnetic dipole effect of the microstrip line feed [12].

$$f_{r,HEM_{11\delta}} = \frac{6.324c}{\pi r\sqrt{\varepsilon_r}}\left\{0.27 + 0.36\left(\frac{r}{h}\right) + 0.02\left(\frac{r}{h}\right)^2\right\}$$

In the above equation, the ‘r’ and ‘h’ is the radius and height of cylindrical ceramic respectively. From the equation, the resonant frequency is found to be 3.35 GHz, which is close to the simulated one.

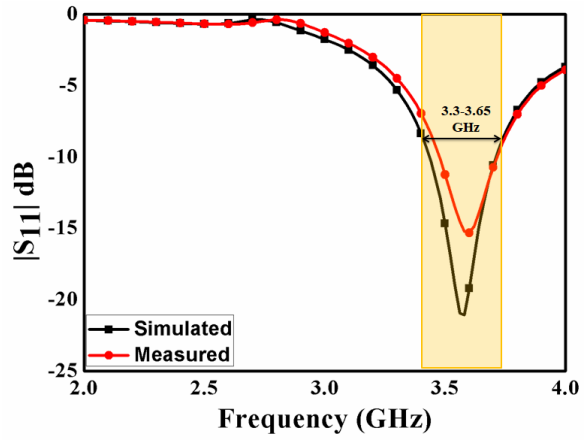


Fig. 6.3 Measured/Simulated $|S_{11}|$ of the proposed antenna

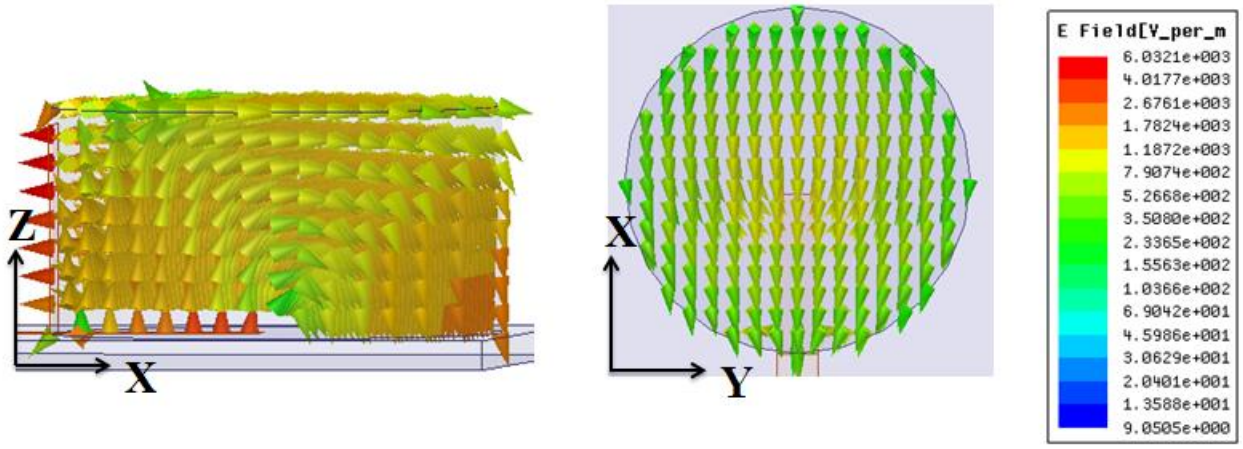


Fig. 6.4 E-field variation of proposed antenna at 3.5 GHz

The suggested antenna's measured/simulated gain fluctuation is shown in Fig. 6.5. With the help of the two-antenna approach [13], gain is measured. Within the operational frequency band, the gain value is around 3.5 dBi, as shown in Fig. 6.5. Fig. 6.6 depicts the proposed antenna's radiation pattern in the XZ and YZ planes at 3.5 GHz. The broadsided radiation pattern is clearly seen in Fig. 6.6, which is owing to the HEM11 mode in CDRA [11]. In both planes, there is a good co-pol to cross-pol disparity.

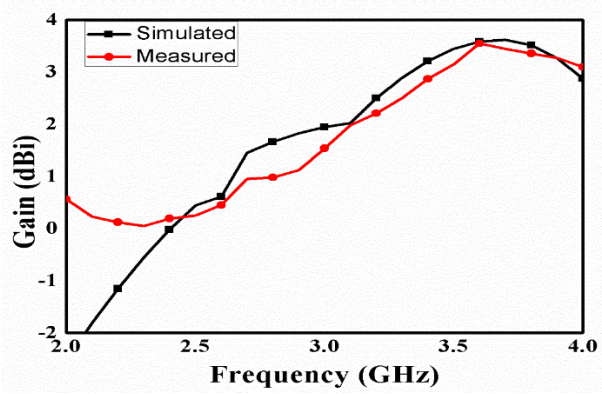


Fig.6.5 Measured/Simulated Gain variation of proposed antenna

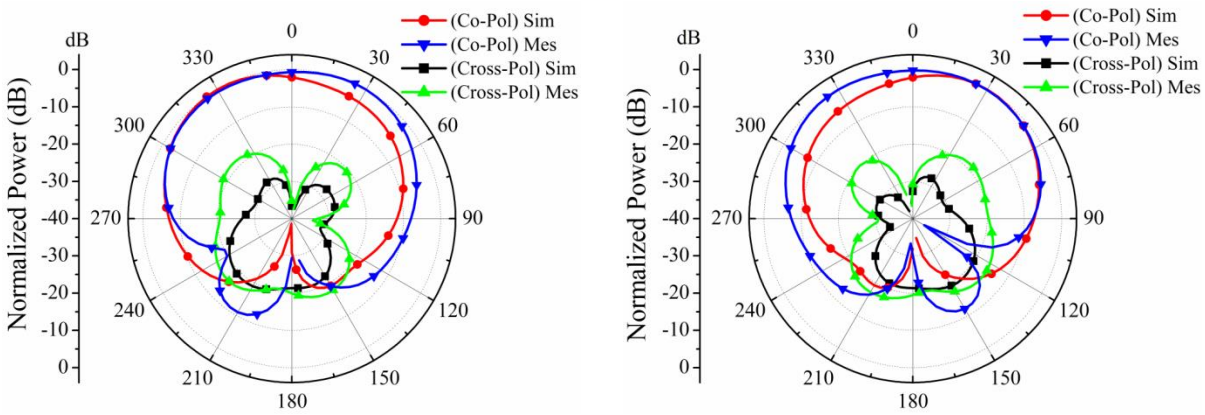


Fig. 6.6 Far-field pattern of proposed antenna at 3.5 GHz (a) XZ plane (b) YZ plane

Conclusion

In this study we went through different Machine Learning techniques which can be leveraged into antenna design optimization. A brief description of each technique was presented followed by their use cases and application. The reference CDRA antenna was then explained with its design characteristics. Later with the HFSS design we tried to optimize the parameters for a specific goal with optimetrics and then with parametrics we generated the training and test data set.

Above mentioned 5 ML techniques were used to firstly train the model followed by tuning each model by analyzing the prediction metrics. Finally for few data samples the predicted values were compared with the HFSS calculated values. This brought us to infer that the prediction performance of all these techniques were quite similar except ANN, despite being from DL domain its model didn't show satisfactory outputs due the small-sized data set and less optimized hyperparameter tuning. With this our study shows the potential and versatility of ML techniques in automated antenna design.

The limitations of the ANN model were addressed by KBNN. We discussed two major techniques of implementation which were source-difference and prior-knowledge-input. With fewer training samples better model accuracy were achieved as evident from the prediction metrics, even the training time of these models were less comparatively. All the work and progress made in the project is thoroughly described in a research-article which is in the process of publication through the Journal of Indian

Academy of Science- Sadhana. Our project's work could then find it's implication in few other important antenna related applications and use-cases.

Future Scope

We carried on our study on a CDRA reference antenna, but the same techniques can be further leveraged by the automated design process of more complicated antenna designs like 3D antennas and a versatile for a versatile range of antennas required in different applications like IoT. As of now just one antenna parameter which is S11 is considered for optimization but probably for better results and optimization other parameters and aspects like the radiation pattern and beam tilting should also be taken into account. A detailed study on several other antenna parameters and their effect in its optimization is something which can be carried on in further iterations by considering and leveraging the results of our study and analysis. Also with automated antenna optimization the fabrication of antenna for different applications could also be done in an automated fashion. These fabricated physical antennas can be used to carry out practical observations to further study the performance of these antennas in real application rather than in HFSS simulation. Training and prediction of these ML models with physical antennas may produce better prediction results as several practical phenomena are taken into account in this case.

References

[1] H. Xin and M. Liang, “3-D-printed microwave and THz devices using polymer jetting techniques,” Proc. IEEE, vol. 105, no. 4, pp. 737–755, Apr. 2017.

[2] A. Massa, G. Oliveri, M. Salucci, N. Anselmi, and P. Rocca, “Learning By-examples techniques as applied to electromagnetics,” J. Electromagn. Waves Appl., vol. 32, no. 4, pp. 516–541, Mar. 2018.

[3] M. Hagan and M. Menhaj, “Training feedforward networks with the Marquardt algorithm,” IEEE Trans. Neural Netw., vol. 5, no. 6, pp. 989–993, 1994.

[4] S. B. Imandoust and M. Bolandraftar, “Application of k-nearest neighbor (kNN) approach for predicting economic events: Theoretical background,” Int. J. Eng. Res. Appl., vol. 3, no. 5, pp. 605–610, Sep./Oct. 2013.

[5] Tianqi Chen and Carlos Guestrin, “XGBoost: A Scalable Tree Boosting System” University of Washington

[6] ANSYS Electromagnetics Suite 15.0, ANSYS, Canonsburg, PA, USA, 2013.

[7] Siyang Sun, Yinghua Lv, Jinling Zhang, Zhidong Zhao, “Optimization based on genetic algorithm and HFSS and its application to the semiautomatic design of antenna”, Article May 2010

[8] P. Watson and K. C. Gupta, EM-ANN models for via interconnects in microstrip circuits, IEEE MTT-S Int Microwave Symp Dig, 1996, pp. 1819]1822.

[9] P. M. Watson, K. C. Gupta, and R. L. Mahajan, Development of knowledge based artificial neural network models for microwave components, IEEE MTT-S Int Microwave Symp Dig, 1998, pp. 9]12.

[10] Rahul Kumar, S. S. Likith Narayan, Somesh Kumar, Sourajeet Roy, Senior Member, IEEE, Brajesh Kumar Kaushik, Senior Member, IEEE, Ramachandra Achar, Fellow, IEEE, and Rohit Sharma, Senior Member, IEEE, “Knowledge-Based Neural Networks for Fast Design Space Exploration of Hybrid Copper-Graphene On-Chip Interconnect Networks”, Article

[11] D. Kajfez, A.W. Glisson, J. James, Computed Modal Field Distributions for Isolated Dielectric Resonators, IEEE Trans. Microw. Theory Tech. 32 (1984) 1609–1616.

[12] R.K. Mongia, P. Bhartia, Dielectric resonator antennas—a review and general design relations for resonant frequency and bandwidth, Int. J. Microw. Millimeter-Wave Comput. Eng. 4 (1994) 230–247

[13] G. A. Stutzman, W.L. and Thiele, Antenna Theory and Design, 2nd ed. Hoboken, NJ, USA: Wiley, 1998