AUTOMATING ANTENNA DESIGN WITH MACHINE LEARNING IN CST

A project report submitted in partial fulfillment of the requirements for the award of the degree of

B.Tech.

in

Electronics and Communication Engineering

By

AMAN KUMAR(621109)

ANIKET KUMAR (621111)

AVANIT KUSHWAHA (621114)

CHIRAG AGARWAL (621133)

RAJ KOHALE (621165)

EC399
MINI PROJECT II
NATIONAL INSTITUTE OF TECHNOLOGY
ANDHRA PRADESH-534101

APRIL 2024

BONAFIDE CERTIFICATE

This is to certify that the project titled **AUTOMATING ANTENNA DESIGN WITH MACHINE LEARNING IN CST** is a bonafide record of the work done by

AMAN KUMAR (621109)

ANIKET KUMAR (621111)

CHIRAG AGARWAL (621133)

AVANIT KUSHWAHA (621114)

RAJ KOHALE (621165)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **ECE** of the **NATIONAL INSTITUTE OF TECHNOLOGY, ANDHRA PRADESH**, during the year 2023-2024.

Dr.A.Arun Kumar	**Dr.S.Yuvaraj**
Project Incharge	Head of the Department

ABSTRACT

Antenna design, a fundamental aspect of modern communication systems, plays a pivotal role in achieving optimal performance in wireless communication networks. Traditional antenna design approaches often rely on analytical and empirical methods, which may be time-consuming and may not fully exploit the design space's complexities. This paper presents a comprehensive review of recent advancements in antenna design facilitated by machine learning (ML) techniques. By harnessing the power of ML algorithms, We have explored novel avenues for optimizing antenna parameters, including radiation pattern, directivity variation with frequency and among others.

The abstract outlines key insights into the integration of ML techniques in antenna design, emphasizing their potential to revolutionize the field. It highlights the advantages of ML-driven approaches, such as improved efficiency, flexibility, and adaptability to diverse design requirements. Furthermore, the abstract discusses various ML algorithms employed in antenna design, like artificial neural networks. Case studies and experimental results are also presented to demonstrate the efficiency and practical implications of ML-enabled antenna design methodologies.

ACKNOWLEDGEMENT

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

Dr. A. ArunKumar, our project incharge, for helping us and guiding us in the course of this project .

Dr. S. Yuvaraj, the Head of the Department, Department of ECE.

Our internal reviewers, **Ch. Chaitanya Krishna** , **Mr. J. Kondalaroa** , **Mrs. J. Dhanashree** for their insight and advice provided during the review sessions.

We would also like to thank our individual parents and friends for their constant support.

TABLE OF CONTENTS

List of Figures

Chapter 1

Introduction

1.1 Introduction

Microstrip patch antennas are widely used due to their compactness and ease of fabrication. However, achieving optimal performance through traditional design methods can be iterative and time-consuming. This project explores the potential of machine learning (ML) to streamline and enhance microstrip patch antenna design.

We propose a methodology where data on frequency variation and directivity for antennas with varying dimensions is collected through simulations in Computer Simulation Technology (CST) software. This rich dataset will then be utilized to train an ML model. The trained model will be empowered to predict optimal antenna geometries that achieve desired frequency and directivity goals. Finally, the project will showcase the capabilities of the ML-designed antenna by simulating its radiation pattern, directivity, and generating a 3D model, providing a comprehensive evaluation of its performance.

This approach holds the promise of significantly accelerating the microstrip patch antenna design process while achieving superior performance characteristics.

Chapter 2

Review Of Literature

2.1 Antenna Design

2.1.1 Microstrip patch antenna

Microstrip patch antennas were first proposed in 1953; however, Munson and Howell developed the first practical Microstrip patch antenna in 1970s. A Microstrip Patch Antenna, in its simplest form consists of a radiating patch on one side of a dielectric substrate, and ground plane on the other side of the substrate. The Microstrip Antennas have numerous advantages over a traditional antenna such as low weight, small size, small volume, and ease of fabrication using printed circuit technology. With increasing requirements for personal and mobile communications, the demand for smaller and low profile, antennas have brought Microstrip patch antennas to the forefront. Apart from having great advantages, it do have some demerits as well such as their low efficiency, low power, high value of Q, poor polarization polarity, poor scan performance, spurious feed radiation and narrow frequency band.

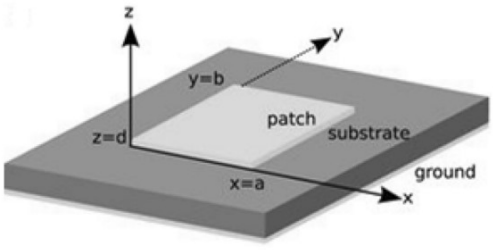

Fig. 1 Structure of a patch antenna

Figure 2.1: Microstrip patch

2.1.2 Feeding techniques for rectangular patch

Microstrip patch antenna has various methods of feeding techniques. As these antennas having dielectric substrate on one side and the radiating element on the other. These feed techniques or methods are being put as two different categories contacting and non-contacting. Contacting feed technique is the one where the power is being fed directly to radiating patch through the connecting element i.e. through the Microstrip line. Non-contacting technique is the one where an electromagnetic magnetic coupling is done to transfer the power between the Microstrip line and the radiating patch. Even though there are many new methods of feed techniques the most popular or commonly used techniques are

1. Microstrip line
2. Coaxial probe
3. Aperture coupling
4. Proximity coupling and
5. Co planar wave guide feed.

1 and 2 being the contacting feed techniques and 3, 4 being non- contacting feed techniques. There are few factors which lead or involve in the selection of a particular type of feed technique. The first and the foremost factor is the efficient power transfer between the radiating structure and the feed structure, i.e. the impedance that is matching between the two. The minimization of the radiation and the effect of it's on the radiation pattern is one of the most important aspect for the evaluation of feed.

In this project we are using Microstrip line feeding technique, a feed point is measured somewhere on the surface of the rectangular patch where the impedance of the patch matches with the impedance of Microstrip feed line that point where the impedance of the printed substrate is approximately 50 Ohms. A coaxial connector is connected perpendicular to the patch surface at this particular point from where the patch antenna can be fed. Below is the coaxial fed structure of an antenna.

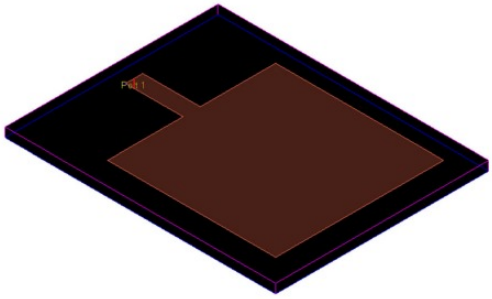

Figure 2.2: Microstrip line feeding

2.2 Data Augmentation

In our scenario, data augmentation aims to artificially increase the size of your dataset (originally 50 rows x 5 features) to 2000 rows x 5 features. This technique helps address the challenge of limited training data, which can hinder the performance of machine learning models.

Common Data Augmentation Techniques

Here are some techniques applicable to various data types (text, images, etc.):

1. Random Noise Injection (Used in the Provided Code): Adds small, random variations to existing features in your dataset. In your case, the code adds values between -0.1 and 0.1 to each feature, introducing slight noise to mimic real-world data variability.

2. Random Cropping (Images):Creates new sub-images by randomly cropping existing images. This can be helpful for tasks like object detection or image classification, where the object of interest might appear at different locations within an image.

3. Flipping (Images):Horizontally or vertically flips images. This can help the model learn to recognize objects regardless of their orientation.

4. Rotation (Images):Rotates images by random angles, enhancing the model's ability to generalize to unseen rotations.

5. Color Jitter (Images):Adjusts brightness, contrast, saturation, and hue of images, increasing the model's robustness to lighting variations.

6. Shifting (Time Series):Shifts time series data points by a random amount along the time axis. This can help the model learn patterns that persist over time, even if they occur at slightly different offsets.

7. Synonym Replacement (Text):Replaces words in text data with their synonyms, creating variations while preserving overall meaning.

8. Random Deletion (Text):Randomly removes words from text data, making the model more resilient to missing or incomplete information.

Applying Data Augmentation to our Dataset

The specific techniques you employ will depend on the nature of your data (50 rows x 5 features). Here's a general idea for potential application (assuming your data is numerical):

1. Random Noise Injection:Implement the approach shown in the provided code, adjusting the noise level (-0.1 to 0.1) as needed based on your data characteristics.

2. Other Techniques (if applicable):If your data allows for techniques like flipping or rotation (e.g., representing images), consider incorporating them while being mindful of the meaning they might convey in your specific domain.

Pros and Cons of Data Augmentation

Pros:

1. Increased Dataset Size:Augmentation allows you to create a larger dataset for training, potentially leading to better model performance and generalization.

2. Reduced Overfitting: A larger and more diverse dataset can help prevent the model from overfitting to the specific training data, improving its ability to handle unseen examples.

3. Improved Model Robustness: By introducing variations, data augmentation can make the model more resilient to noise and slight changes in the input data.

Cons:

1. Potential for Unrealistic Data:Depending on the technique and its implementation, augmentation might create data points that are unrealistic or nonsensical in your domain. It's crucial to choose techniques that yield valid augmentations relevant to your task.

2. Increased Computational Cost: Processing and training with a larger augmented dataset can require more computational resources and time.

In Conclusion

Data augmentation is a valuable technique for addressing limited training data and enhancing model performance. However, it's essential to select and implement augmentation methods that generate realistic and relevant variations for your specific data and machine learning problem.

Chapter 3

Proposed work with its methodology

3.1 Antenna Design

3.1.1 Antenna Analysis and Simulation

The initial stage involves analyzing and simulating candidate antenna designs using CST Studio. This powerful software allows us to model the electromagnetic behavior of various antenna geometries. By defining material properties, dimensions, and excitation ports, we can virtually simulate the performance characteristics of the antennas, such as radiation patterns, gain, return loss, and bandwidth. This simulation helps identify promising antenna designs for further investigation.

3.1.2 Data Collection and Preparation

The next step involves collecting necessary information and data from the CST simulations. This data typically includes key design parameters (independent variables) that influence the antenna's performance and the corresponding performance metrics (dependent variables) obtained from the simulations (e.g., gain, directivity,operating frequency). This data will be used to train the machine learning model.

3.2 ML Model Design

3.2.1 Data Augmentation

In cases where the initial dataset obtained from simulations is limited (less than 50 samples), data augmentation techniques can be employed. These techniques artificially

expand the dataset by creating new data points from existing ones through methods like random rotations, scaling, or adding controlled noise. This helps the model generalize better and avoid overfitting to the limited training data. Through data augmentation, we aim to increase the dataset size to a more robust level for training (e.g., 2000 rows).

3.2.2 Development of an ANN-based Regression Model

An Artificial Neural Network (ANN) based regression model serves as the core of this methodology. This model will be trained on the prepared data, learning the relationship between the design parameters (input) and the resulting antenna performance (output). The model architecture will likely consist of an input layer with a number of neurons equal to the number of design parameters (e.g., 2 features), a hidden layer with a moderate number of neurons (e.g., 64) to capture the complex relationships, and an output layer with a number of neurons corresponding to the desired performance metrics (e.g., 3 outputs for gain, return loss and bandwidth). The training process involves iteratively adjusting the weights and biases within the network to minimize the error between the model's predictions and the actual performance data obtained from simulations.

3.2.3 Automated Antenna Design with Model Generated Parameters

Once trained, the ANN model can act as a design optimization tool. By feeding new design parameter values into the input layer, the model will predict the corresponding antenna performance. This allows for an automated design exploration process. We can utilize the generated design parameters within an automated scripting environment within CST Studio. This script can then create new antenna geometries based on the model's suggestions, eliminating the need for manual design iterations.

3.2.4 Structure of our ANN model

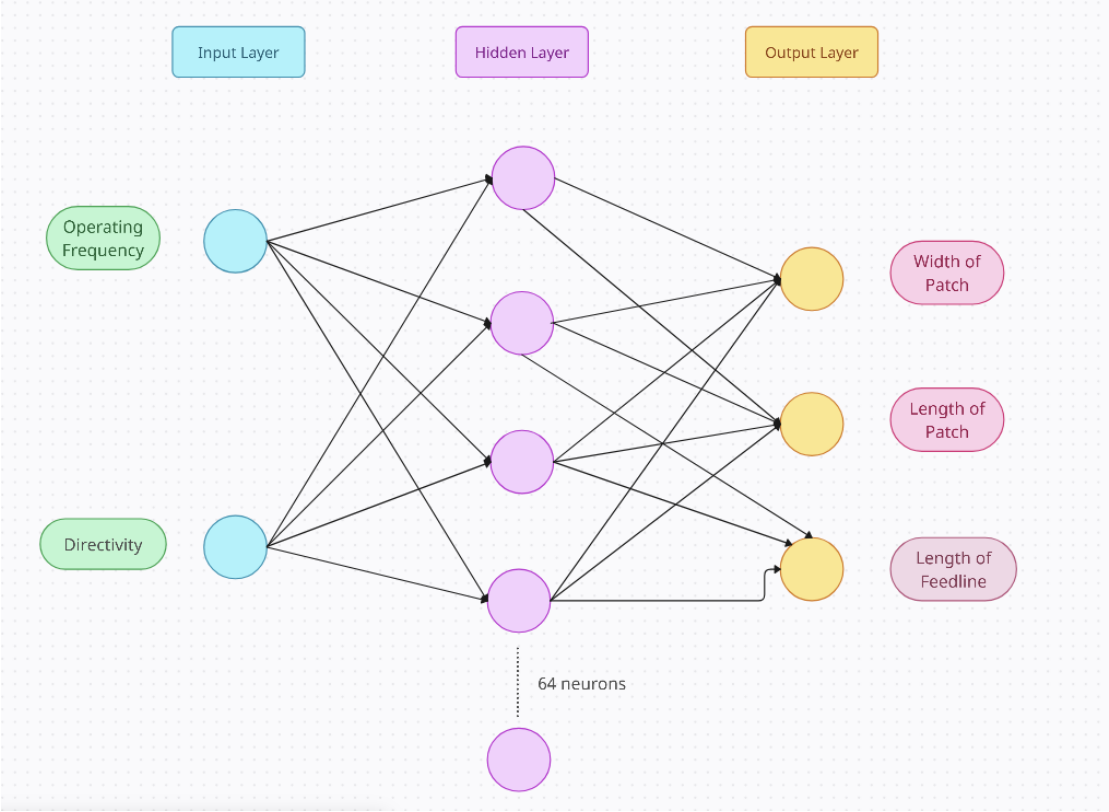

Figure 3.1: ANN Model

3.3 Flowchart of methodology

This methodology offers a data-driven approach for antenna design and optimization. It leverages the power of CST Studio simulations to generate initial data and utilizes machine learning to discover the underlying relationships between design parameters and antenna performance. This allows for efficient exploration of design space and the creation of high-performance antennas with minimal manual effort.

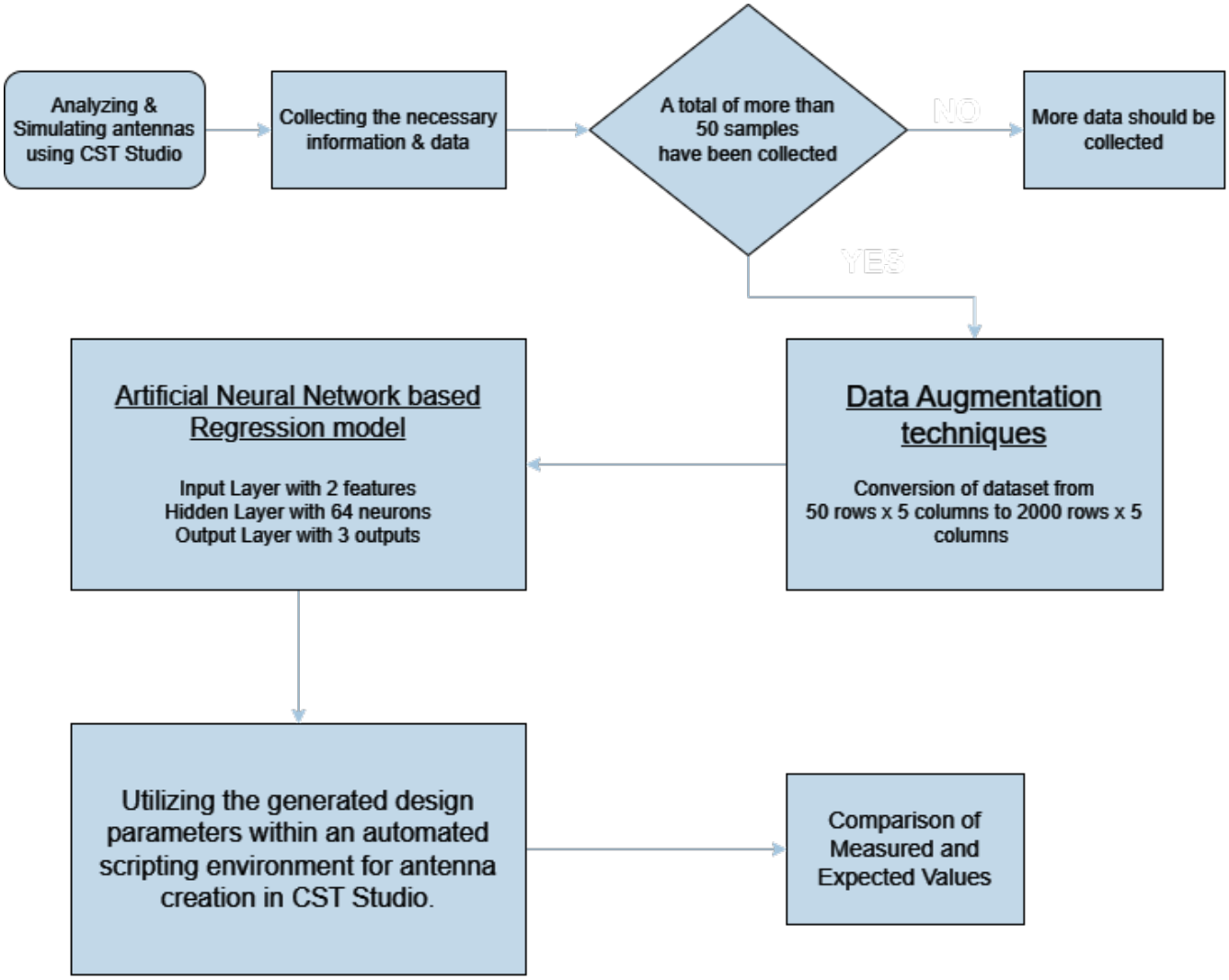

Figure 3.2: Flowchart

Chapter 4

Results and Discussion with Future Scope

4.1 Simulated results

4.1.1 3D microstrip patch antenna

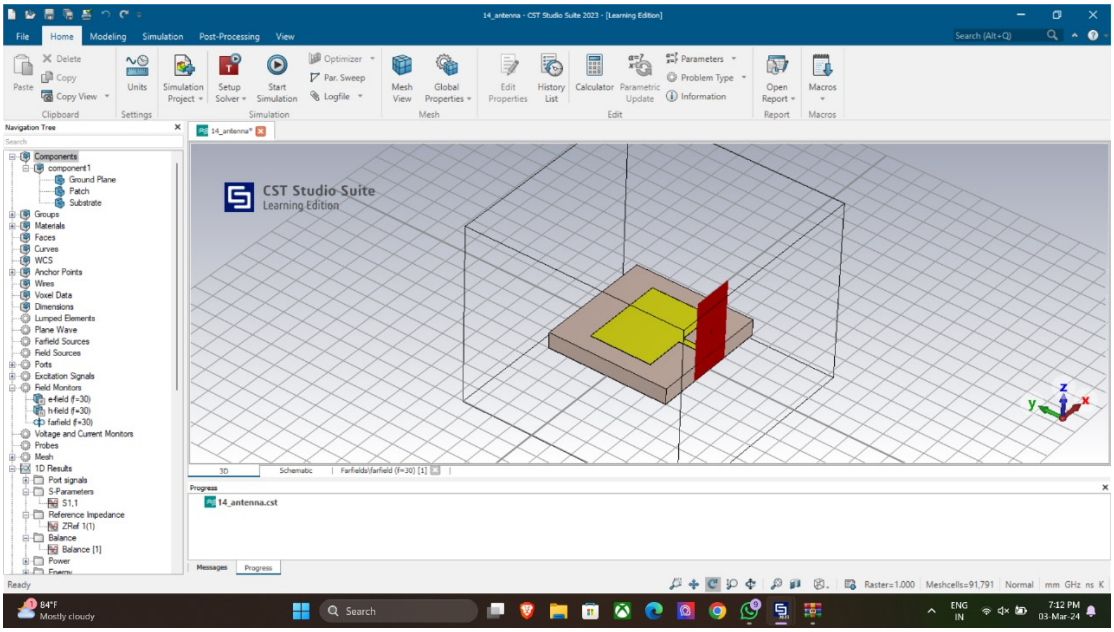

Figure 4.1: Microstrip line feeding

4.1.2 Radiation pattern

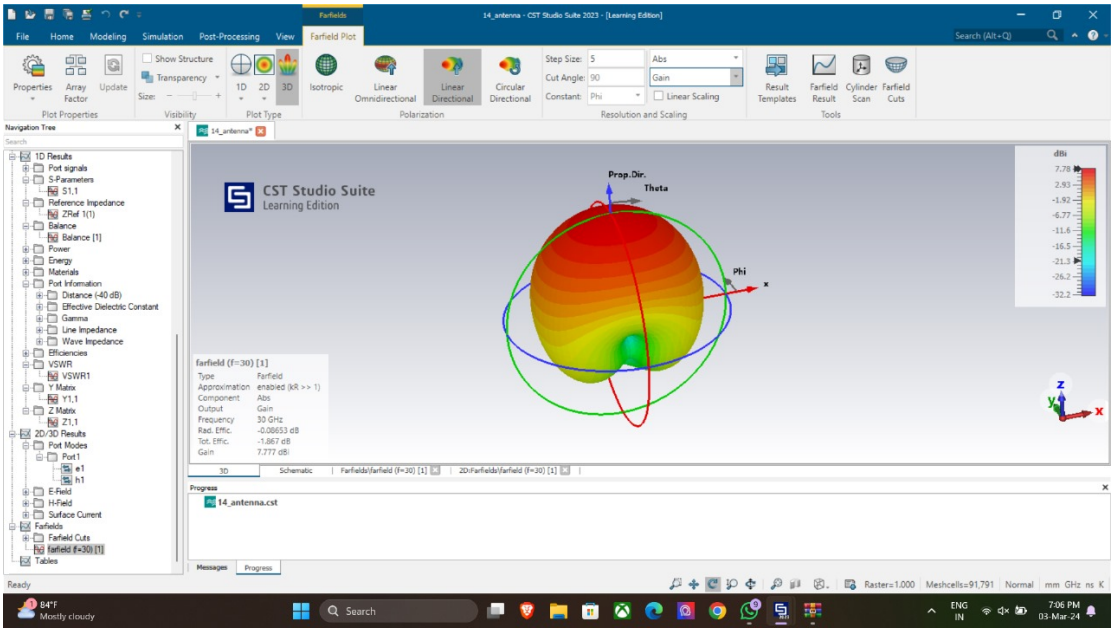

Figure 4.2: Radiation Pattern

4.1.3 S parameters vs Frequency variation

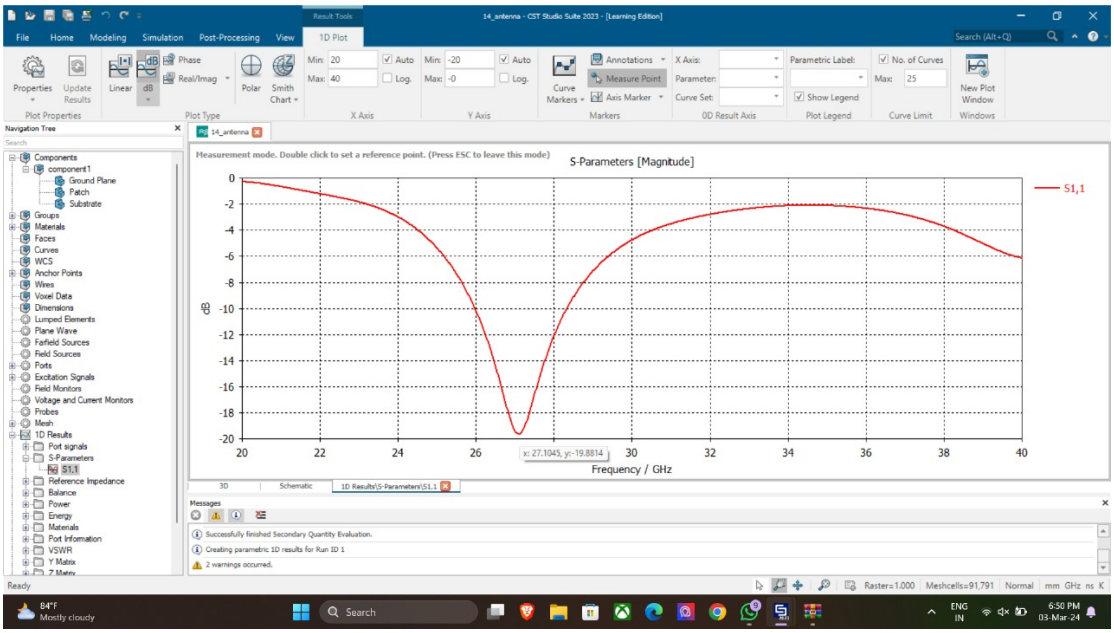

Figure 4.3: S parameters vs frequency

4.1.4 H field Variation

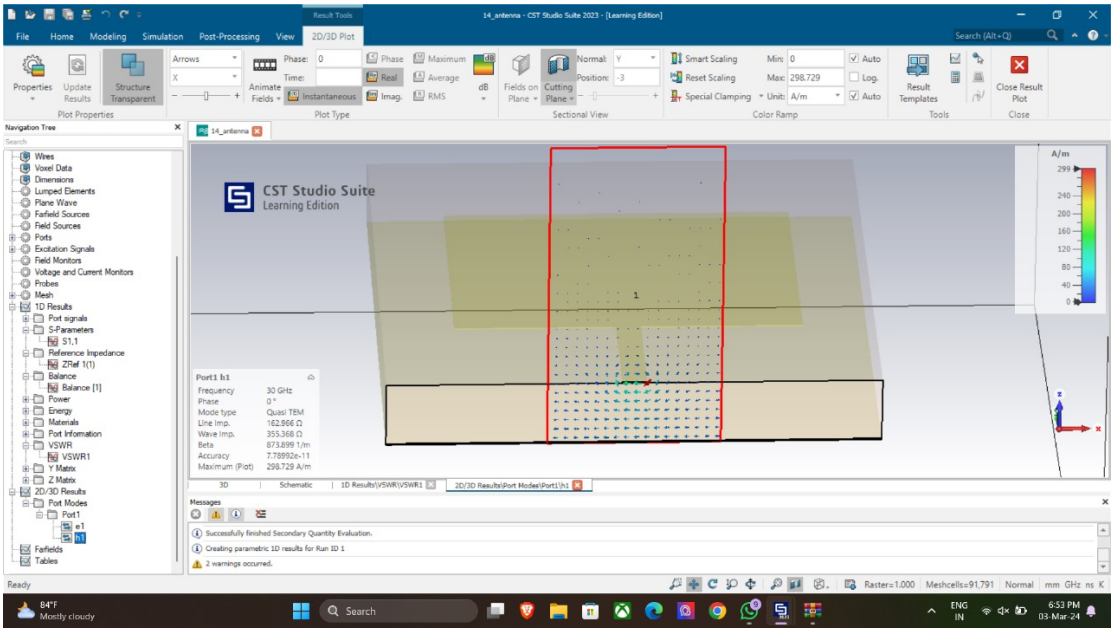

Figure 4.4: H field Variation

4.1.5 ML output result

```
dir = float(input("ENTER THE VALUE OF DIRECTIVITY (in DB):\t"))
opF = float(input("ENTER THE VALUE OF OPERATING FREQ. (in GHz):\t"))

result = (model.predict([[dir,  opF]]))

print("--------------------RESULTS---------------------")
print("\n\n(width of patch):   ", result[0,0])
print("(length of patch):   ", result[0,1])
print("(length of feedline):   ", result[0,2])
```

```
ENTER THE VALUE OF DIRECTIVITY (in DB): -1.29
ENTER THE VALUE OF OPERATING FREQ. (in GHz):    98
1/1 [==============================] - 0s 40ms/step
--------------------RESULTS---------------------


(width of patch):    1.034759
(length of patch):    0.44232625
(length of feedline):    2.788932
```

Figure 4.5: ML model output

4.2 Future Scopes

Machine learning (ML) is poised to significantly impact the future of microstrip patch antenna (MPA) design. Traditionally, this process involves time-consuming simulations and intricate calculations. However, ML can analyze vast datasets to predict optimal antenna geometries that achieve desired performance metrics like resonance frequency, bandwidth, and gain. This streamlines the design process and reduces development time. Furthermore, ML can handle the complexities of multi-objective optimization, where conflicting performance parameters need to be balanced. This allows for the creation of antennas that excel in multiple aspects simultaneously. Beyond optimizing existing designs, ML opens doors for exploring unconventional antenna shapes and configurations, potentially leading to entirely new functionalities or miniaturized sizes.

As new materials and fabrication techniques emerge, ML can predict their impact on antenna performance, enabling the design of MPAs specifically tailored to these advancements. Finally, machine learning models can integrate with 3D printing and other additive manufacturing techniques, facilitating the creation of antennas with intricate geometries that are challenging to produce with traditional methods. However, for ML to reach its full potential, interpretable models are crucial. This ensures engineers understand the reasoning behind the model's recommendations and fosters trust in its capabilities. Additionally, the success of ML-based design relies heavily on the quality and quantity of training data. Open sharing of antenna design data can significantly accelerate progress in this exciting field.

4.3 Conclusion

In conclusion, this project demonstrates the significant potential of machine learning in transforming microstrip patch antenna design. By leveraging ML's capabilities, we can look forward to faster development cycles, enhanced antenna performance, and innovative antenna functionalities for next-generation wireless communication systems.

Bibliography

[1] Hector Kaschel and Cristian Ahumada. Design of rectangular microstrip patch antenna for 2.4 ghz applied a wban. In *2018 IEEE International Conference on Automation/XXIII Congress of the Chilean Association of Automatic Control (ICA-ACCA)*, pages 1–6, 2018.

[2] John Colaco and Rajesh Lohani. Design and implementation of microstrip patch antenna for 5g applications. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pages 682–685, 2020.

[3] Hilal M. El Misilmani and Tarek Naous. Machine learning in antenna design: An overview on machine learning concept and algorithms. In *2019 International Conference on High Performance Computing Simulation (HPCS)*, pages 600–607, 2019.

[1] `https://ieeexplore.ieee.org/document/8609703`

[2] `https://ieeexplore.ieee.org/document/9137921`

[3] `https://ieeexplore.ieee.org/document/9188224`

Appendices

Appendix A

Code Attachments

A.1 Code for Data Augmentation

```
import random
import pandas as pd

def augment_data(data, num_augmentations):
  """
  Augments the data by adding a random noise to each feature.

  Args:
      data: A DataFrame containing the original data.
      num_augmentations: The number of augmented data points to
    generate.

  Returns:
      A DataFrame containing the original data and the augmented data
    points.
  """
  augmented_data = []
  for _ in range(num_augmentations):
    # Select a random data point from the original data
    original_point = data.sample()  # Use pandas.DataFrame.sample()

    # Create a copy to avoid modifying the original data
    augmented_point = original_point.copy()

    # Add random noise to each feature (adjust noise level as needed)
    for col in augmented_point.columns:
      augmented_point[col] += random.uniform(-0.1, 0.1)  # Adjust
    noise range

    # Append the augmented point as a DataFrame
    augmented_data.append(augmented_point)

  return pd.concat([data] + augmented_data, ignore_index=True)  #
    Concatenate with original data

# Load your original data from a CSV file (replace 'your_data.csv'
    with your actual file path)
data = pd.read_csv('your_data.csv')

```

```
# Increase data size to 1000 rows using augmentation
augmented_data = augment_data(data, 950)  # Generate 950 new data points (adjust as needed)

# Use the augmented_data for your machine learning tasks
print(augmented_data.head())  # Print the first few rows of the augmented data
import csv
import numpy as np
from tensorflow import keras
from sklearn.model_selection import train_test_split  # Import for train-test split
import pandas as pd


# Load your CSV data (replace 'data.csv' with your actual file path)
data = pd.read_csv('/content/augmented_data (1).csv')


"""# Separate inputs (X) and outputs (y)
X = [[record[4], record[5]] for record in data]
y = [[record[1], record[2], record[3]] for record in data]
"""

X = data.iloc[:,3:]
y = data.iloc[:,0:3]
# Convert data to NumPy arrays
X = np.array(X)
y = np.array(y)

# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

A.2 Code for ML Model

```
import csv
import numpy as np
from tensorflow import keras
from sklearn.model_selection import train_test_split  # Import for train-test split
import pandas as pd


# Load your CSV data (replace 'data.csv' with your actual file path)
data = pd.read_csv('/content/augmented_data (1).csv')


"""# Separate inputs (X) and outputs (y)
X = [[record[4], record[5]] for record in data]
y = [[record[1], record[2], record[3]] for record in data]
"""

X = data.iloc[:,3:]
y = data.iloc[:,0:3]
# Convert data to NumPy arrays
```

```
X = np.array(X)
y = np.array(y)

# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

 # Define the model
model = keras.Sequential([
  keras.layers.Dense(16, activation='relu', input_shape=(2,)),  # Input layer with 2 features
  keras.layers.Dense(64, activation='relu'),                    # Hidden layer with 32 neurons
  keras.layers.Dense(3)                                          # Output layer with 3 outputs
])

# Compile the model with optimizer and loss function (mean squared error for regression)
model.compile(optimizer='adam', loss='mse')

# Train the model (replace with your training data)
model.fit(X_train, y_train, epochs=150)

# Make predictions on new data
y_pred = model.predict(X_test)
dir = float(input("ENTER THE VALUE OF DIRECTIVITY (in DB):\t"))
opF = float(input("ENTER THE VALUE OF OPERATING FREQ. (in GHz):\t"))

result = (model.predict([[dir,  opF]]))

print("----------------------RESULTS----------------------")
print("\n\n(width of patch):   ", result[0,0])
print("(length of patch):   ", result[0,1])
print("(length of feedline):   ", result[0,2])
```