Pixelated Antenna Design Using Machine Learning

Atharva Tiwari

June 28, 2024

1 Introduction

The advancement of wireless technology in telecommunications and IoT heavily depends on the design and optimization of antenna systems. This project aims to create a more generalized form of pixelated antenna, improving upon conventional design techniques through iterative modifications and using active learning and bayesian optimization to improve and fast the model training.

2 Design Evolution

2.1 Initial Design

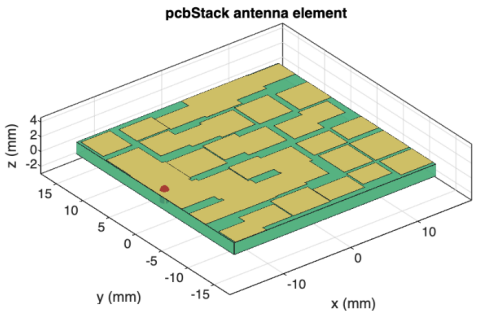

Figure 1: Illustration of the pixelated antenna design.

Our initial design concept involved a 6x6 pixel grid, with each pixel containing 36 rectangular patches. The length and width of each patch were chosen randomly from a specified range, and the center was offset from the corresponding pixel's center by a specified amount. The parameters for each patch were stored for further analysis.

2.2 First Modification

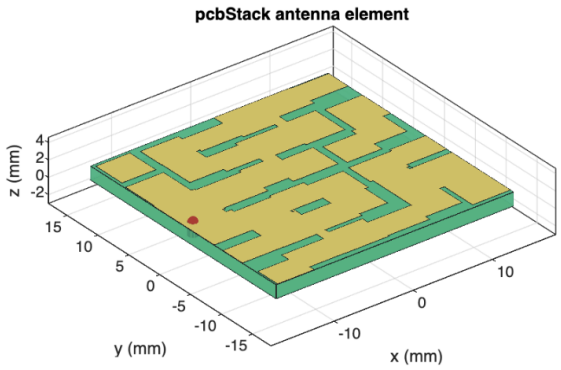

Figure 2: Illustration of the pixelated antenna design.

The design was modified to join rectangular patches that were very close to each other if the distance between them was less than a threshold value. This helped in reducing unnecessary divisions and merging closely located patches.

2.3 Final Modification

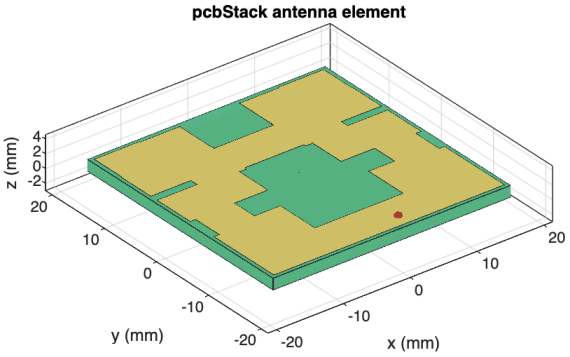

Figure 3: Illustration of the final modified design of the pixelated antenna.

To further refine the design and reduce metal content, we constructed 18 rectangular patches for each pixel on the left half of the board. The length

and width of each patch were chosen randomly from a specified range, and the center was offset similarly. Close patches were merged by increasing their length and shifting them. Additionally, a specified percentage of patches were randomly removed. The bottom row was ensured to be fully metal to maintain feed connectivity. The right half of the board was then created as a mirror image of the left half.

3 Optimization and Parameter Tuning

We conducted extensive research and experiments to determine the optimal parameters for our design, aiming for a good percentage of resonant structures evenly spread between 2-6 GHz. The parameters found to be optimal were:

- Pixel size: 5.75mm
- Grid: 6x6
- Threshold: 0.5mm
- Range of length of patch: 0.85-1.2 times pixel size
- Range of width of patch: 0.85-1.2 times pixel size
- Offset range for center's x and y-coordinates: -0.1 to +0.2 times pixel size

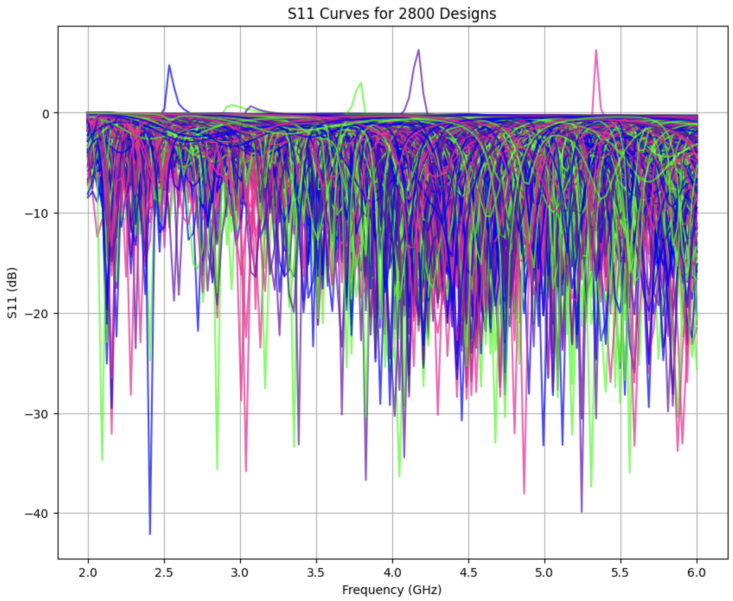

Figure 4: S11 curves for all the samples.

For these parameters, we simulated 2800 samples and obtained resonance for 2071 samples, achieving a 73% resonance rate, which is significantly higher than the 20-40% achieved by other pixelated design techniques. You can see the same in the plot above.

4 Dataset Generation

The input dataset `patchcoords.txt` contains the features/parameters for every patch for each design. The output dataset `sparameters.txt` contains S11 values for each design at 128 frequencies, equispaced between 2-6 GHz.

5 Machine Learning Models

5.1 Feedforward Neural Networks and 1D CNN

Initially, we trained simple feedforward neural networks and 1D CNNs on the data. Various combinations of convolutional layers, fully connected layers, and dropouts were tried, but the best mean squared error (MSE) achieved was 5.19 due to the complexity of the data and limited dataset size.

5.2 3D CNN

We then tried a 3D CNN approach, dividing the data into four channels corresponding to four features, with a height and width of 6 and a depth of 1. Despite trying techniques like normalizing input data, introducing regularizations, and making the model smaller, significant improvement was not achieved. The training loss improved, but the validation loss remained in the range of 5-6. Training the model on 1000 datasets and 2800 datasets reduced the MSE from 5.74 to 5.26. The current architecture used for the 3D CNN is:

5.2.1 Network Architecture

The architecture of the PatchAntennaCNN3D model is structured as follows:

1. **Conv1** → **BatchNorm1** → **ReLU** → **Dropout1**
2. **Conv2** → **BatchNorm2** → **ReLU** → **Dropout2**
3. **Flatten** the output from **Conv2**.
4. **Fully Connected Layer 1** → **ReLU**
5. **Fully Connected Layer 2** (**Output**)

5.2.2 Detailed Layer Descriptions

- **Conv1**: Convolutional layer with 16 filters of size (1, 3, 3), stride (1, 1, 1), and padding (0, 1, 1).
- **BatchNorm1**: Batch Normalization layer for the output of Conv1.
- **ReLU**: Rectified Linear Unit activation function.
- **Dropout1**: Dropout layer with a probability of 0.2.
- **Conv2**: Convolutional layer with 32 filters of size (1, 3, 3), stride (1, 1, 1), and padding (0, 1, 1).
- **BatchNorm2**: Batch Normalization layer for the output of Conv2.
- **Dropout2**: Dropout layer with a probability of 0.3.
- **Flatten**: Flatten the output tensor from Conv2 to prepare it for the fully connected layers.
- **Fully Connected Layer 1**: Dense layer with 256 units.
- **Fully Connected Layer 2**: Dense layer with 128 output units corresponding to the 128 frequency points.

6 Active Learning

6.1 Polygon Dataset

6.1.1 Introduction to Active Learning

Active learning is a machine learning approach where the model is allowed to query the most informative data points for labeling. This method is particularly useful when labeled data is scarce or expensive to obtain. By focusing on the most informative samples, active learning aims to improve model performance with fewer labeled instances compared to random sampling.

6.1.2 Active Learning Techniques and Their Theory

- **Bayesian Optimization (EI, PI, UCB, LCB)**:
  - **Expected Improvement (EI)**: Selects the sample that maximizes the expected improvement over the current best observation. The improvement $I(x)$ at a point x is defined as:

    $$I(x) = \max(0, f(x) - f(x^+))$$

    where $f(x^+)$ is the current best observation.

  - **Probability of Improvement (PI)**: Selects the sample with the highest probability of improving upon the current best observation. The probability of improvement at a point x is given by:

$$PI(x) = P(f(x) > f(x^+))$$

  - **Upper Confidence Bound (UCB)**: Balances exploration and exploitation by considering both the mean and uncertainty of predictions. The UCB at a point x is:

$$UCB(x) = \mu(x) + \kappa\sigma(x)$$

    where $\mu(x)$ is the predicted mean, $\sigma(x)$ is the predicted standard deviation, and κ is a parameter balancing exploration and exploitation.
  - **Lower Confidence Bound (LCB)**: Similar to UCB but focuses on minimizing the lower bound of the confidence interval. The LCB at a point x is:

$$LCB(x) = \mu(x) - \kappa\sigma(x)$$

- **Query By Committee (QBC)**: Uses multiple models (the committee) to evaluate the uncertainty of predictions. Variants include:
  - **Variance**: Measures the disagreement among committee members. The variance of predictions at a point x is:

$$\text{Var}(x) = \frac{1}{M}\sum_{i=1}^{M}(\hat{y}_i(x) - \bar{y}(x))^2$$

    where $\hat{y}_i(x)$ is the prediction of the i-th committee member and $\bar{y}(x)$ is the mean prediction.
  - **KL Divergence**: Measures the divergence between the probability distributions predicted by committee members. The KL divergence at a point x is:

$$KL(p\|q) = \sum_i p(x_i)\log\frac{p(x_i)}{q(x_i)}$$

    where p and q are the predicted distributions.

- **Bayesian Active Learning by Disagreement (BALD)**: Selects samples that maximize the mutual information between predictions and model parameters. The mutual information $I(y, \theta|x)$ is:

$$I(y, \theta|x) = H(y|x) - E_{p(\theta|D)}[H(y|x, \theta)]$$

  where H is the entropy.

- **Expected Model Change Maximization (EMCM)**: Selects samples that are expected to induce the largest change in the model parameters upon retraining. The expected model change at a point x is:

$$\Delta\theta = E[\|\theta' - \theta\||x]$$

  where θ' are the parameters after retraining.

6.1.3 Comparison of Active Learning Techniques

The following table summarizes the differences between the active learning techniques in terms of informativeness, diversity, and representativeness:

Technique	**Informativeness**	**Diversity**	**Representativeness**
Bayesian Optimization (EI)	High	Medium	Low
Bayesian Optimization (PI)	High	Medium	Low
Bayesian Optimization (UCB)	Medium	High	Medium
Bayesian Optimization (LCB)	Medium	High	Medium
QBC (Variance)	High	High	Medium
QBC (KL Divergence)	High	High	Medium
BALD	High	Medium	Medium
EMCM	High	Low	Medium

Table 1: Comparison of active learning techniques

6.1.4 Results on Polygon Datset

We applied active learning on a polygon dataset, initializing 100 labeled indices using k-means clustering and using various techniques like Bayesian optimization (EI, PI, UCB, LCB), QBC, BALD, and EMCM. The results were as follows:

Training and Selecting Techniques	**MSE**
On total train dataset (2848 samples)	2.13
Random/Passive Sampling (1500 samples)	2.59 (avg of 10 times random results)
Bayesian Optimization with EI (1500 samples)	3.21
Bayesian Optimization with UCB (1500 samples)	3.1688
Bayesian Optimization with LCB (1500 samples)	2.293
Bayesian Optimization with PI (1500 samples)	2.315
QBC with Variance (1500 samples)	2.2464
QBC with KL Divergence (1500 samples)	2.415
Bayesian Active Learning with Disagreement (1400 samples)	2.17
Expected Model Change Maximization (1400 samples)	3.06

Table 2: Results of various active learning techniques on the polygon dataset

We can see that Bayesian Active Learning with Disagreement performs bets in this case and gives us loss comparable to training on whole dataset in only half samples.

In this only qbc,emcm have been implemented with k-means clustering initialization. Other strategies we experimeted by ourseleves.

6.2 Design Dataset

For the design dataset, we used QBC with variance, Bayesian optimization with LCB, and BALD, achieving MSEs of 5.209, 5.19, and 5.265 respectively on 1000

samples. In this case Bayesian optimization performs best with lcb as aquistion function.

7 Conclusion

This project has successfully developed a generalized pixelated antenna design and applied machine learning techniques for optimization. Despite challenges in reducing validation loss, the results indicate significant progress. There needs to be extensive work done on model to improve it by changing its architecture but we have proof of concept for our active learning techniques.