

UC Irvine

UC Irvine Previously Published Works

Title

Optimization of Antenna Array Configurations Using Deep Learning

Permalink

<https://escholarship.org/uc/item/8zh6z1fh>

Journal

IEEE Open Journal of Antennas and Propagation, 6(5)

ISSN

2637-6431

Authors

Lu, David Lin Yi

Maman, Lior

Earls, Jackson

et al.

Publication Date

2025-10-01

DOI

10.1109/ojap.2025.3578887

Copyright Information

This work is made available under the terms of a Creative Commons Attribution License, available at <https://creativecommons.org/licenses/by/4.0/>

Peer reviewed

Optimization of Antenna Array Configurations Using Deep Learning

David Lin Yi Lu ¹, Lior Maman ², Jackson Earls ¹, Amir Boag ², FELLOW, IEEE, and Pierre Baldi ¹, FELLOW, IEEE

¹Department of Computer Science, University of California Irvine, Irvine, CA 92617, USA

²School of Electrical and Computer Engineering, Tel Aviv University, Tel Aviv 69978, Israel

CORRESPONDING AUTHOR: Amir Boag (e-mail: boag@tau.ac.il).

The work was partially supported by Israel Ministry of Innovation, Science and Technology under Grant 7037.

ABSTRACT We introduce a deep learning-based optimization method that enhances the design of sparse phased array by reducing grating lobes. Our approach begins with a generation of sparse antenna array configurations, efficiently addressing the non-convex challenges and high degrees of freedom in array design. We then employ neural networks, trained on 70,000 and tested on 30,000 configurations, to approximate a non-convex cost function that measures the ratio between the energy of the main lobe and the side lobe level. The approximation is differentiable and allows for the minimization of the cost function by gradient descent with respect to the antenna's elements coordinates, yielding a new optimized configuration. A custom penalty mechanism is also implemented, integrating various physical and design constraints into our optimization method. The effectiveness of our method is tested on the ten configurations with the lowest initial cost values, showing a further reduction in cost by 699% to 816%, with an average of 740%. The code for the method is available at <https://github.com/david5010/Optimization-of-Antenna-Arrays>.

INDEX TERMS Antenna Arrays, Machine Learning, Deep Learning, Gradient Descent, Optimization

I. Introduction

Phased array systems employing sparse configurations are often challenged by the emergence of grating lobes, primarily due to constraints on the maximum number of elements and beamwidth specifications that dictate the size of the antenna aperture. These lobes are typically a consequence of the periodic arrangement found in conventional antenna arrays [1]. To mitigate this, in most cases sparse arrays avoid periodic layouts in favor of aperiodic or randomized element distributions [2], [3], which help suppress grating lobes by breaking the spatial regularity that leads to their formation. However, designing aperiodic arrays introduces significant hurdles, including a substantial increase in the degrees of freedom that must be optimized and the considerable computational complexity involved in assessing the antenna radiation patterns across all main beam scan angles. Traditional methods falter in reliably identifying global optima due to the problem's complex landscape. The grid search approach, which refers to a uniform and exhaustive sampling of the solution space across all configuration parameters, becomes impractical due to the combinatorial explosion of possible antenna array configurations and the

substantial computational resources required to evaluate each one. Several approaches to optimize the thinned and the sub-array configurations have been proposed in literature, incorporating various advanced techniques to refine and enhance antenna design [4]–[10]. In addition, previous approaches specifically addressing sparse arrays have utilized stochastic optimization techniques, such as simulated annealing or genetic algorithms, to design aperiodic linear sparse arrays to minimize side lobe levels and optimize array configurations [11]. While these methods are flexible and can handle complex, non-linear optimization problems, they are often computationally intensive and slow, especially when optimizing large arrays. For sparse arrays, the domain of possible configurations is enormous, as each element can be shifted differently. Even with a coarse discretization allowing K possible shift per element, the total number of configurations becomes $K^{N_{\text{ele}}}$, where N_{ele} is the number of elements. This exponential growth renders the optimization process highly challenging even for stochastic optimization methods.

Machine learning methodologies have been increasingly utilized in recent years to optimize complex problems in

the field of electromagnetics (EM). In particular, the use of neural networks has proven to be an effective and time-efficient method of optimization. These networks have achieved significant success in modeling computationally intensive EM simulations [12], [13] and complex EM relationships [14], [15] while maintaining a high degree of accuracy. Beamforming, a method of signal processing, has been the focus of numerous deep learning applications [16]–[19]. While conventional beamforming techniques reduce side lobe levels (SLLs) by modifying the excitation of individual antenna elements, our method maintains uniform excitation and instead focuses on optimizing the positions of the elements. In contrast to approaches that feed a desired array pattern into a neural network [19], our model takes a predefined sparse array configuration as input and predicts its corresponding cost. This estimated cost guides an iterative refinement process, where gradient descent is applied directly to the element coordinates with respect to the cost function.

We address the challenges of computational speed and complexity by first reducing the number of variables in the phased array scanning problem from five variables (the observation vector, the scanning vector, and the wave number) to two [20]–[23]. We then generate sparse array data using a sub-array approach [24], decreasing the number of freedom in the optimization process. From this generated data, we select array configurations that demonstrate promising results and further refine these configurations using a deep learning approach. By employing neural networks, we accurately approximate the complex, non-convex cost function of an array, enabling efficient minimization of the cost through gradient descent on the antenna array coordinates.

II. Methods

A. Generation of Antenna Array Configurations

In this paper, we focus on the transmission mode case. To achieve maximum power, thermal efficiency, and overall efficiency, most solid-state amplifiers are operated in saturation mode. This, in turn, imposes a constraint of uniform amplitude across the array.

In this section, we address the Active Electronically Scanned Array (AESA) design challenge by simplifying the AESA array factor (AF) characterization from five to two variables and describe the process of data generation for optimization. For a planar phased array antenna with uniformly excited elements located in the yz -plane, the AF is given by:

$$U(k, AZ, EL, AZ_0, EL_0) = \sum_{n=1}^{N_e} e^{jk\mathbf{r}_n \cdot (\hat{\mathbf{r}} - \hat{\mathbf{r}}_0)} \quad (1)$$

Here, k is the wave-number, $\hat{\mathbf{r}} = \cos(EL)\sin(AZ)\hat{\mathbf{y}} + \sin(EL)\hat{\mathbf{z}} + \cos(EL)\cos(AZ)\hat{\mathbf{x}}$ represents the unit vector in the observation direction, $\mathbf{r}_0 = \cos(EL_0)\sin(AZ_0)\hat{\mathbf{y}} + \sin(EL_0)\hat{\mathbf{z}} + \cos(EL_0)\cos(AZ_0)\hat{\mathbf{x}}$ denotes the main beam

direction. Also, EL and AZ denote the elevation and azimuth of the direction of observation, respectively, while EL_0 and AZ_0 pertain to the main beam direction. Note that $EL = \pi/2 - \theta$ and $AZ = \varphi$ of the conventional spherical coordinates system. The $\mathbf{r}_n = y_n\hat{\mathbf{y}} + z_n\hat{\mathbf{z}}$ indicates the location of the n th element and N_e is the number of elements. By introducing a difference vector $\mathbf{u} = k(\mathbf{r} - \mathbf{r}_0)$, we can redefine the array factor using only two variables:

$$U(u_y, u_z) = \sum_{n=1}^{N_e} e^{j(y_n u_y + z_n u_z)} \quad (2)$$

u_y and u_z are the y and z components of $\mathbf{u}$. The domain of $\mathbf{u}$ can be determined numerically or analytically given the main beam scan sector. Identifying the AF's support within the u_y, u_z plane simplifies the evaluation over the entire domain of interest. This method reduces the resources needed for storage and processing, making it easier to characterize the entire observation area across the complete scan sector in a single illustration.

To streamline the optimization process, simplify the search domain, and facilitate the generation of both well- and poorly-performing arrays for a diversified dataset, we fixed the antenna size and adopted the concept of *local periodicity* for data generation [24]. This principle facilitates the organization and integration of radiating elements and their associated transmit/receive modules by employing a sub-array strategy. In this method, the entire array is constructed from multiple sub-arrays, each designed to exhibit periodicity, thereby significantly reducing the number of design variables.

In practical terms, the array surface is segmented into distinct subdomains, with each region hosting a periodic sub-array. These sub-arrays are designed with flexibility in both positioning and orientation, allowing for variations in periodicity along the x and y axes. To mathematically formalize the sub-array configuration, we divide the full array into N_{groups} sub-arrays. Each sub-array is then rotated by an angle selected from a predefined rotation vector:

$$\text{Angle} = [0, \Delta\theta, 2\Delta\theta, \dots, \Theta_{\max} - \Delta\theta], \quad (3)$$

where $\Delta\theta = \Theta_{\max}/N_{\text{groups}}$. We vary the parameter $\Theta_{\max}$ within the range $[0^\circ, 90^\circ]$, and for each value, we randomly assign rotation angles to the sub-arrays by permuting the *Angle* vector. This ensures each sub-array receives a distinct orientation, introducing structural diversity while keeping the search space manageable.

This approach simplifies the otherwise complex task of array design by reducing the multitude of potential configurations to a structured yet flexible set of variables. The key parameters for optimization include: periodicity along each axis, rotation angle, and sub-array position.

For example:

- When $\Theta_{\max} = 0^\circ$, all sub-arrays remain unrotated, resulting in a completely aligned configuration. For an

inter-element spacing of 2λ , this setup typically yields high side-lobe levels (SLL), making it a representative case of poor array performance.

- When $\Theta_{\max} = 90^\circ$, and each group is randomly rotated using the full rotation vector, the structural randomness generally leads to improved performance. In certain permutations, this can achieve SLL levels as low as -13 dB, similar to that of a standard rectangular array.

In our dataset, we fixed the antenna aperture size to 64λ , with a maximum of 32 elements along each axis. This allowed us to generate both high- and low-performing array examples, making the dataset ideal for training deep learning models to estimate cost functions while reducing the risk of overfitting.

Figure 1 illustrates an example of an array configuration utilizing periodic sub-arrays with $\Theta_{\max} = 90^\circ$.

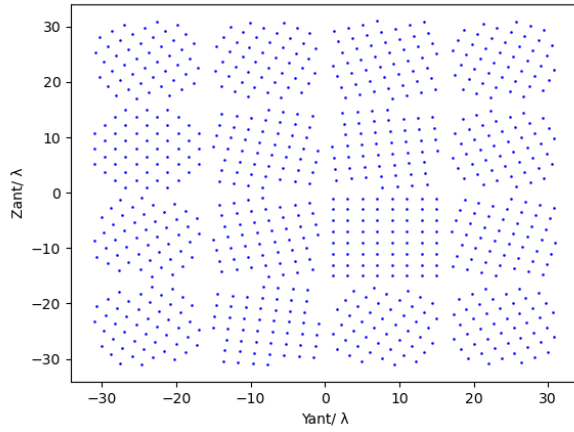


FIGURE 1. Example of a periodic sub-array configuration with $\Theta_{\max} = 90^\circ$.

B. Cost Function

The cost function evaluates the ratio of the main lobe (ML) to the side-lobe level (SLL) while adhering to the constraints

$$f_{\text{cost}} = - \frac{\sum_{i \in ML} |AF(u_y(i), u_z(i))|^{2p}}{\sum_{i \notin ML} |AF(u_y(i), u_z(i))|^{2p}} \quad (4)$$

where p determines the norm type, and i indices the samples in the $\mathbf{u}$ -plane. Taking the norm to infinity, i.e., using the ℓ_∞ norm, corresponds to penalizing only the maximum value. This is effective for suppressing prominent grating lobes but tends to under-penalize patterns with many smaller sidelobes, resulting in noisy radiation patterns. On the other hand, setting $p = 1$ corresponds to taking the average absolute value across the pattern. While this provides a more global assessment, it significantly reduces the influence of high peaks and therefore fails to sufficiently penalize sharp and undesired sidelobes in the radiation pattern. As a compromise, we set $p = 4$ for our data generation. This choice imposes stronger penalties on significant peaks while

still accounting for the overall sidelobe structure. It ensures that excessively high sidelobes, such as grating lobes, are discouraged from appearing outside the main lobe region.

III. Optimization of Antenna Arrays using Neural Network Approximations

Optimizing antenna array configurations directly presents a significant challenge, particularly when the objective involves minimizing a complex and potentially non-differentiable cost function. The complexity of the true cost function necessitates extensive engineering efforts to optimize directly. To circumvent these challenges, we employ neural networks as a strategic means to approximate the true cost function accurately, serving as a surrogate cost function. It has been shown that neural networks have the universal approximation property [25]–[28]. Neural networks, by design, provide a smooth and continuous function. This characteristic enables the application of gradient descent methods for optimization. The choice of surrogate function is highly

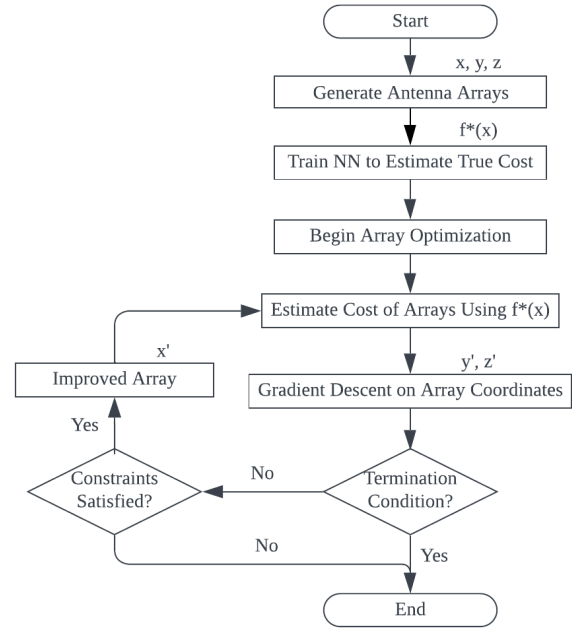


FIGURE 2. Flowchart illustrating the antenna array optimization process, outlining steps from initial configuration through iterative adjustments using gradient-based optimization and constraint mechanisms.

flexible, provided it accurately approximates the original cost function. This surrogate can take various architectural forms, such as feed forward neural networks (FNNs), convolutional neural networks (CNNs), or transformer variants, as the primary optimization process occurs after a reliable estimate is obtained. In our experiments, we tested an FNN and a Set Transformer [29] as surrogate functions. We chose the FNN for its simplicity and computational efficiency, and the Set Transformer for its ability to handle set-based inputs and

maintain permutation invariance. Once we derive an accurate approximation of the cost function using the FNN and Set Transformer models, we begin the optimization process. The optimization process begins by inputting a given sparse array configuration to the surrogate deep learning model. Next, an approximation of the cost associated with the inputted array is produced by the model. We then perform gradient descent directly on the antenna u_y, u_z coordinates to produce a new array with a better resulting cost. This process is conducted over a set number of iterations until the cost converges and an optimal configuration is reached. Figure 2 illustrates this method.

A. Model Selection

For optimal model architecture and hyperparameter identification, we utilized SHERPA, a tool for hyperparameter optimization [30]. We conducted a grid search for each adjustable parameter in both the Feedforward Neural Network (FNN) and Set Transformer models.

1) Feedforward Neural Network (FNN) Model

The architecture of the FNN model comprises four fully connected layers. The initial input layer is set to a dimension of 2048, representing the Y and Z coordinates of 1024 array elements. This is followed by two hidden layers with dimensions of 20 and 12, respectively. Nonlinearity is introduced after each hidden layer through the application of ReLU activation functions. The final output layer contains a single neuron designed to predict the cost related to the specified antenna array geometry.

During the training phase, the FNN underwent 1000 epochs with a learning rate of 1×10^{-5} and a batch size of 128, using the Adam optimizer to reduce the loss function.

Various configurations were tested, including first and second hidden layer dimensions ranging from [4, 120] and [2, 40] neurons, respectively. Learning rates were explored from 1×10^{-2} to 1×10^{-6} , and batch sizes varied from 16 to 256 to establish the most effective training parameters.

2) Set Transformer

The Set Transformer is engineered to process set-structured data, as described by Lee et al. [29]. It is structured with an encoder and a decoder. The encoder consists of two permutation equivariant set attention blocks to ensure order invariance of inputs, while the decoder includes a multi-head attention block for pooling, two set-attention blocks, and a fully connected output layer.

Both the encoder and decoder feature attention blocks with two heads each, and the hidden layers within maintain a dimensionality of 32. The model accepts inputs with a dimension of 2, corresponding to the Y and Z coordinates of each antenna element, and the output layer, similar to the

FNN, has a single neuron that computes the cost for the antenna array configuration.

Training of the Set Transformer was carried out over 1000 epochs with a learning rate of 1×10^{-3} , a batch size of 64, and employing the Adam optimizer.

We also tested various configurations for the Set Transformer, adjusting hidden layer dimensions from 8 to 64 and attention head counts from 2 to 8. Additionally, learning rates ranged from 1×10^{-2} to 1×10^{-6} , and batch sizes were experimented with from 16 to 256.

B. Cost

To estimate the cost function depicted in (4), the neural network takes an inputted array configuration and outputs a real number, where a lower cost indicates better performance. Our generated dataset consists of arrays with initial cost values ranging from -40,000 to -0.7. Due to the wide range of output values, we normalized these costs for training via (5). Once trained, the model outputs a normalized cost value for the inputted array configuration. For interpretability, denormalization is applied to the outputs to recover the original cost values using (6). In (5) and (6), J represents the cost.

$$J_{\text{norm}} = \frac{J - \mu_J}{\sigma_J} \quad (5)$$

$$J = (J_{\text{norm}} \cdot \sigma_J) + \mu_J \quad (6)$$

Given that our dataset was generated based on the same parameters (same wavelength and antenna aperture size), the coordinates of the array elements will determine the cost of the configurations. Each array is represented as a set of coordinate pairs with varying numbers of elements.

C. Inputs

The primary challenge with input handling arises from the chosen neural network architecture. For architectures sensitive to input order and length variations, such as the FNN in our case, we address this by applying a consistent ordering to the antenna coordinates across all arrays. Additionally, we pad each input to match the maximum number of antennas in the dataset (1024) to ensure uniform input length.

Once the neural network provides a reliable estimate of the cost function, the model serves as the foundation for optimizing antenna configurations. By representing the configurations in a structured input format, we facilitate the use of gradient-based optimization methods to refine antenna positions.

To optimize a given antenna array configuration, we adjust the coordinates of the elements to locate positions that minimize the cost function. Using the neural network surrogate within a PyTorch framework [31], [32], we employ the Adam optimizer, a gradient-based optimization technique known for its computational efficiency and tunable hyperparameters [33].

To further refine the optimization process, we incorporate mechanisms that maintain physical constraints among the array elements. The first approach is a minimum distance check, where each candidate configuration is evaluated at each iteration to ensure that elements adhere to a set minimum distance constraint. If the candidate configuration meets this constraint, it is accepted, and the optimization proceeds to the next iteration. If not, the configuration reverts to the last valid arrangement, ending the optimization.

The second approach involves integrating a repulsion term as a penalty function within the cost model. This term imposes a penalty based on the proximity of element pairs, allowing the optimization to continue without interruption. The penalty is structured to increase significantly as any pair of elements approaches the minimum distance threshold, effectively discouraging configurations that violate spacing requirements. Designed to asymptotically approach infinity as the distance nears the limit, this penalty can also be customized to discourage other behaviors, such as elements spreading out excessively or moving inward, ensuring that all configurations remain within acceptable bounds throughout the optimization process.

D. Penalty Mechanism

In our model, the penalty mechanism is adaptable, allowing for the implementation of custom penalties based on specific design requirements. These penalties can be formulated to address various constraints or optimization objectives. The example below illustrates the penalty function we employed.

This function calculates the Euclidean pairwise distance between each pair of elements, denoted as D_{ij} for elements i and j , where i and j are $1, 2, \dots, N_e$. A minimum threshold is also defined as θ .

$$P_{ij} = \log \left(\frac{1}{D_{ij} - \theta} \right) \quad (7)$$

The total penalty P applied during the optimization is then the sum of all individual penalties, scaled by a factor ϵ , which dictates the repulsion term's influence on the optimization process:

$$P = \epsilon \cdot \sum_{\substack{\forall i \in \text{Ant} \\ \forall j \in \text{Ant}}} P_{ij} \quad (8)$$

Here, Ant represents the set of elements in the configurations. The scalar ϵ is pivotal as a higher value places greater emphasis on maintaining distance between elements, effectively preventing them from nearing the minimum threshold.

The optimization process aims to minimize the custom loss function defined as

$$\text{loss} = f(\text{Ant}) + P \quad (9)$$

where $f(\text{Ant})$ is the output of the surrogate cost network for a given antenna configuration, and P is our custom penalty term.

IV. Results

In this section, we present the performance enhancements of the antenna arrays achieved using the FNN and Set Transformer, both with and without the penalty mechanism intended to reduce mutual coupling. We used here $\lambda = 1$ and set the maximum scan angle to 20° in both elevation and azimuth axes. Figure 3 illustrates the optimized sparse array obtained by the FNN when the penalty mechanism isn't applied. Figure 4 provides an example of an optimized antenna configuration obtained when the penalty mechanism is incorporated. Figure 5 depicts the outcomes for both the FNN and Set Transformer without the penalty mechanism, whereas Figure 6 compares the FNN results with and without the penalty in place. To streamline the analysis, we selected the ten antenna array configurations with the lowest initial costs. Each optimization run required approximately one minute to complete. Our approach is inherently scalable, as multiple optimization tasks can be executed in parallel.

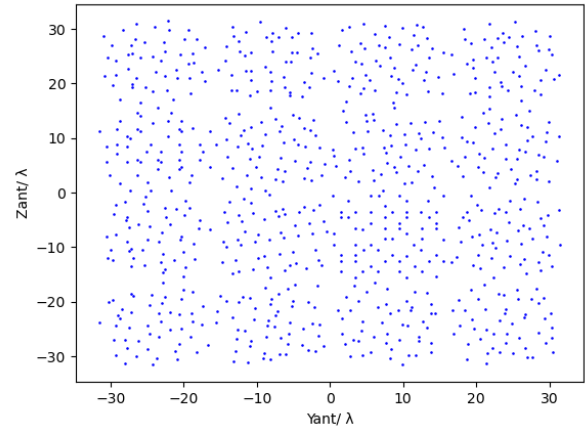


FIGURE 3. Optimized array configuration using the FNN model without the penalty mechanism.

The results show that the half-power beamwidth remains consistently stable ($\zeta_{3\text{dB}} = 0.78^\circ$), where ζ represents either EL or AZ across all configurations, except for the FNN with the penalty mechanism, which achieved a lower value ($\zeta_{3\text{dB}} = 0.70^\circ$). This is reasonable, as observed in Fig. 4, where the antenna array becomes larger. For a detailed analysis of the SLL, we evaluate both the first and the second SLL. The FNN with the penalty mechanism delivers the best performance, achieving first SLL values of $(-17.37, -16.80)$ dB and second SLL values of $(-19.56, -19.99)$ dB. In comparison, the Set Transformer and the FNN without the penalty mechanism yield first SLL values of $(-14.42, -14.06)$ dB and $(-13.59, -13.44)$ dB, respectively, and second SLL values of $(-15.27, -14.93)$ dB and $(-14.29, -14.84)$ dB. The values in parentheses correspond to measurements in the u_y and u_z directions, respectively. It is important to note that the optimization results consistently yield SLL across all scanning angles, with the maximum scan angle

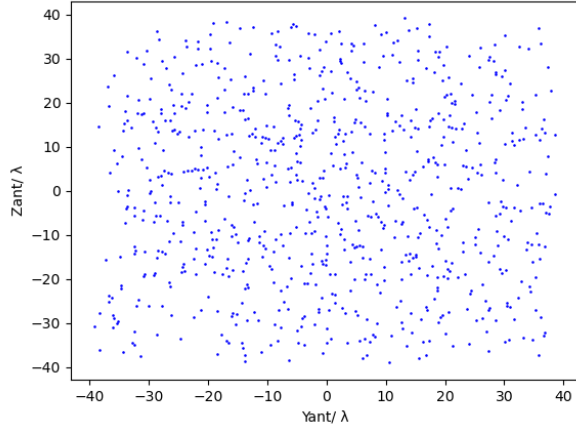


FIGURE 4. Optimized array configuration using the FNN model with the penalty mechanism.

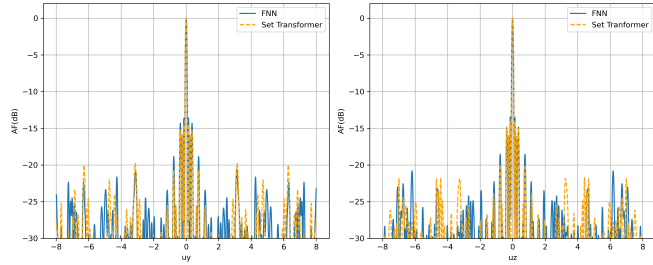


FIGURE 5. u_y and u_z cuts are obtained using both the FNN and the Set Transformer. Solid lines represent results from the FNN, while dashed lines denote those obtained with the Set Transformer.

set to 20° , which covers the intended scanning domain. This demonstrates the main advantage of formulating the problem in the u_y and u_z domains.

The FNN architecture that utilizes the penalty mechanism achieves superior performance. One strategy to reduce mutual coupling involves increasing the minimum distance between the nearest elements, which expands from an average minimum distance of 0.501λ in optimizations without the penalty mechanism to 0.510λ when the penalty mechanism is implemented.

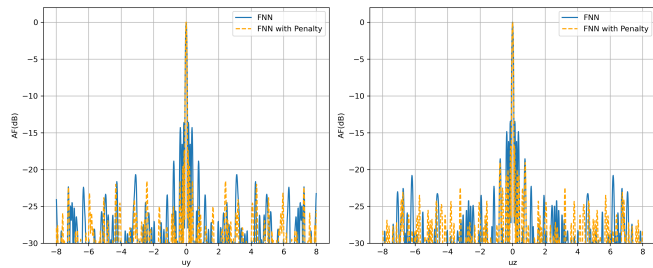


FIGURE 6. u_y and u_z cuts are obtained using the FNN with and without the penalty mechanism. Solid lines represent results from the FNN without penalty, while dashed lines denote those obtained with penalty.

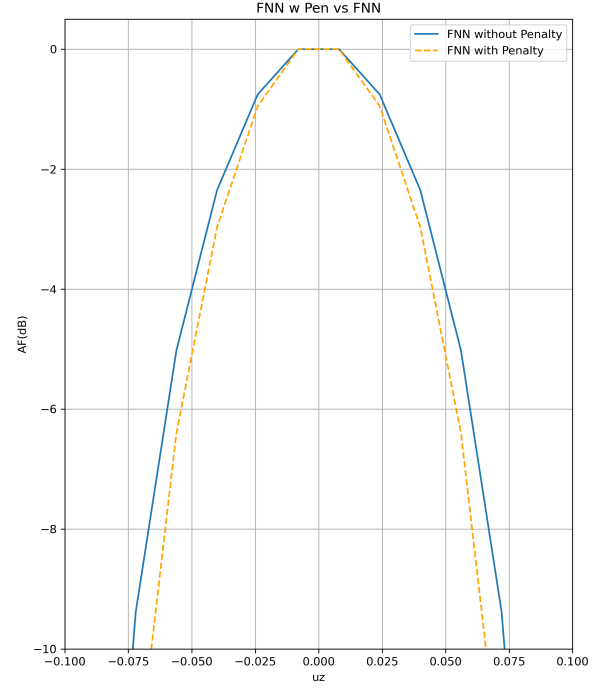


FIGURE 7. A comparison between FNN models with and without a penalty mechanism focused ML area in the antenna pattern.

A. Impact of Gradient Descent without Penalty

An analysis of the 10 lowest cost configurations without using the custom penalty revealed an average cost reduction of 77%, with the most notable improvement observed at 87%.

TABLE 1. Comparison of Cost by Optimization without Penalty

Config.	FNN			SetTrans.	
	Cost Bef.	Cost Aft.	%Chg	Cost Aft.	%Chg
0	-40463.98	-70656.40	-74.61%	-50062.60	-23.72%
1	-39931.30	-74892.03	-87.55%	-48461.69	-21.36%
2	-39767.27	-70558.84	-77.42%	-47678.69	-19.89%
3	-39472.41	-71855.93	-82.04%	-47099.10	-19.32%
4	-39431.64	-71640.69	-81.68%	-52461.51	-33.04%
5	-39383.22	-69344.87	-76.07%	-48599.66	-23.40%
6	-39036.94	-71497.47	-83.15%	-49166.52	-25.94%
7	-38702.13	-64275.86	-66.07%	-49435.75	-27.73%
8	-38690.57	-67786.53	-75.20%	-69324.78	-79.17%
9	-38600.70	-66521.94	-72.33%	-51598.96	-33.67%

A closer examination of the minimum distances between elements post-optimization reveals that the distances often approached the threshold ($\theta = 0.5$) closely. This proximity to the threshold suggests that the optimization process seeks to minimize costs, but it does so at the risk of compromising the physical feasibility and operational integrity of the antenna

arrays by pushing the configurations to the limits of the minimum distance constraint.

B. Effectiveness of the Penalty Function

The application of gradient descent, implementing a penalty for the minimum distance constraint, has demonstrated substantial improvements in the cost of antenna array configurations. With our custom penalty mechanism, we observed an average improvement of 740% across the configurations examined, with the largest improvement being 816% for the FNN model. These results demonstrate the potential of

TABLE 2. Comparison of Cost by Optimization with Penalty

Config.	Cost Bef.	FNN		SetTrans.	
		Cost Aft.	%Chg	Cost Aft.	%Chg
0	-40463.98	-352208.37	-770.42%	-72366.55	-78.84%
1	-39931.30	-322455.62	-707.52%	-68401.11	-71.29%
2	-39767.27	-323139.62	-712.57%	-67188.80	-68.95%
3	-39472.41	-327770.81	-730.37%	-70953.76	-79.75%
4	-39431.64	-361288.12	-816.23%	-67442.47	-71.03%
5	-39383.22	-327317.71	-731.10%	-68203.04	-73.17%
6	-39036.94	-344308.65	-782.00%	-92805.77	-137.73%
7	-38702.13	-323761.15	-736.54%	-77390.64	-99.96%
8	-38690.57	-309294.50	-699.40%	-46534.00	-20.27%
9	-38600.70	-317903.28	-723.56%	-78371.62	-103.03%

gradient descent to effectively refine antenna array designs to achieve significant performance enhancements.

Optimization with a penalty function demonstrates the ability to guide the optimization process within the bounds of operational feasibility while still maximizing cost reduction. Array optimization with penalty ensures that improvements in antenna array configurations do not compromise practical applicability. Furthermore, the variability introduced by adjusting ϵ underscores the nuanced control that the penalty function offers over the optimization process. This control allows for a tailored balance between aggressive cost reduction and adherence to physical limitations. We found that a ϵ value of 12.5 and 1 proved to be most optimal for the FNN and Set Transformer architectures, respectively. These ϵ values were obtained by testing ϵ in the range of 0.1 to 100.

C. Computational Speed and Complexity

The computational overhead of the proposed method is low. We evaluated the computational complexity of the optimal FNN model, which comprises 41.24×10^3 parameters and requires approximately 41.24×10^3 multiply-accumulate (MAC) operations. On an NVIDIA RTX 4070 ti GPU, the model achieves an inference time of approximately 0.25 ms per sample. All experiments were conducted using PyTorch 2.5.1.

D. Comparative Analysis

When comparing the outcomes of optimization processes, we see the importance of our surrogate cost function and the relationship between cost reduction and minimum distance between elements.

The FNN, having an MSE of 0.007, outperformed the Set Transformer on the majority of configurations with an MSE of 0.06. This shows that a closer approximation to the true cost function allows for more accurate gradients, resulting in a larger cost reduction and obtaining better SLLs.

Array optimization with the penalty function not only maintained a safer distance from the threshold but also performed better than the array optimization without penalty. This improvement was observed using both the FNN and Set Transformer models. This optimization method ensures that the configurations remain within acceptable limits, thereby enhancing the practical viability and efficiency of the optimized antenna arrays.

This comparative analysis underscores the importance of maintaining compliance with essential design constraints. The use of a custom penalty term within the optimization process proved to be substantially more effective than optimization without it. Employing this custom penalty term resulted in both greater cost reduction and more significant spacing between array elements.

V. Conclusion

We have introduced a new design methodology for antenna arrays, utilizing deep learning to optimize existing large antenna configurations efficiently. This approach begins with designing sub-arrays as the fundamental basis for configuring the entire array. The core of our method is the application of deep learning, employing neural networks to drive subsequent refinements. With a deep learning approach, we have shown an efficient way to optimize large antenna arrays while significantly improving the cost and having flexibility in defining design and physical constraints. Using deep learning, we reduced the costs of the ten best antenna array configurations by an average of 740% with penalty and 77% without penalty. This new approach, which uses neural networks to estimate the cost function, has the potential to be applied in future work for a variety of applications, such as phase tapering, beam shaping methods, and thinned array design, among others.

REFERENCES

- [1] N. Amitay, V. Galindo, and C. P. Wu, *Theory and Analysis of Phased Array Antennas*. New York: Wiley-Interscience, 1972.
- [2] D. Kurup, M. Himdi, and A. Rydberg, "Synthesis of uniform amplitude unequally spaced antenna arrays using the differential evolution algorithm," *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 9, pp. 2210–2217, 2003.
- [3] L. H. Gabrielli and H. E. Hernandez-Figueroa, "Aperiodic antenna array for secondary lobe suppression," *IEEE Photonics Technology Letters*, vol. 28, no. 2, pp. 209–212, 2016.
- [4] P. Rocca, G. Oliveri, R. J. Mailloux, and A. Massa, "Unconventional phased array architectures and design methodologies—a review," *Proceedings of the IEEE*, vol. 104, no. 3, pp. 544–560, 2016.

- [5] P. Rocca, R. Mailloux, and G. Toso, "Ga-based optimization of irregular subarray layouts for wideband phased arrays design," *IEEE Antennas and Wireless Propagation Letters*, vol. 14, pp. 131–134, 2014.
- [6] P. Rocca, R. L. Haupt, and A. Massa, "Sidelobe reduction through element phase control in uniform subarrayed array antennas," *IEEE Antennas and Wireless Propagation Letters*, vol. 8, pp. 437–440, 2009.
- [7] P. Rocca, L. Manica, and A. Massa, "An improved excitation matching method based on an ant colony optimization for suboptimal-free clustering in sum-difference compromise synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 57, no. 8, pp. 2297–2306, 2009.
- [8] R. L. Haupt, *Antenna arrays: a computational approach*. John Wiley & Sons, 2010.
- [9] R. L. Haupt, "Thinned arrays using genetic algorithms," *IEEE Transactions on Antennas and Propagation*, vol. 42, no. 7, pp. 993–999, 1994.
- [10] R. L. Haupt, "Interleaved thinned linear arrays," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 9, pp. 2858–2864, 2005.
- [11] A. Trucco and V. Murino, "Stochastic optimization of linear sparse arrays," *IEEE Journal of Oceanic Engineering*, vol. 24, no. 3, pp. 291–299, 1999.
- [12] Y. Kim, S. Keely, J. Ghosh, and H. Ling, "Application of artificial neural networks to broadband antenna design based on a parametric frequency model," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 3, pp. 669–674, 2007.
- [13] X. Ding, V. Devabhaktuni, B. Chattaraj, M. Yagoub, M. Deo, J. Xu, and Q. J. Zhang, "Neural-network approaches to electromagnetic-based modeling of passive components and their applications to high-frequency and high-speed nonlinear circuit optimization," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, no. 1, pp. 436–449, 2004.
- [14] K.-C. Lee, "Application of neural network and its extension of derivative to scattering from a nonlinearly loaded antenna," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 3, pp. 990–993, 2007.
- [15] J. Liu, Z.-X. Chen, W.-H. Dong, X. Wang, J. Shi, H.-L. Teng, X.-W. Dai, S. S. T. Yau, C.-H. Liang, and P.-F. Feng, "Microwave integrated circuits design with relational induction neural network," *arXiv preprint arXiv:1901.02069*, 2019.
- [16] H. Huang, Y. Peng, J. Yang, W. Xia, and G. Gui, "Fast beamforming design via deep learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 1065–1069, 2020.
- [17] T. Lin and Y. Zhu, "Beamforming design for large-scale antenna arrays using deep learning," *IEEE Wireless Communications Letters*, vol. 9, pp. 103–107, Jan. 2020.
- [18] S. Komeyliian, "Deep neural network modeling of different antenna arrays; analysis, evaluation, and application," *IEEE Canadian Journal of Electrical and Computer Engineering*, vol. 44, no. 3, pp. 261–274, 2021.
- [19] X. Yang, D. Yang, Y. Zhao, J. Pan, and Y. Chen, "Synthesis of linear sparse array using dnn-based machine-learning method," *IEEE Transactions on Antennas and Propagation*, vol. 71, no. 8, pp. 6513–6522, 2023.
- [20] L. Maman, S. Zach, and A. Boag, "Phase-only beam broadening for array antennas," in *2023 17th European Conference on Antennas and Propagation (EuCAP)*, pp. 1–3, IEEE, 2023.
- [21] L. Maman, S. Zach, and A. Boag, "Beam broadening by phase tapering in wideband electronically scanned arrays," in *2021 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS)*, (Tel Aviv, Israel), Nov. 2021.
- [22] L. Maman, S. Zach, and A. Boag, "Beam broadening by phase tapering in wideband electronically scanned arrays," in *2023 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS)*, (Tel Aviv, Israel), Jul. 2024.
- [23] L. Maman, S. Zach, and A. Boag, "Beam shaping by phase-only waveform encoding for transmitting array antennas in radar applications," *IEEE Open Journal of Antennas and Propagation, Early Access*, 2025.
- [24] A. Boag, S. Zach, Y. Gazit, G. Zwirn, and A. Boag, "Grating lobe mitigation in ultra-wideband phased arrays," in *2018 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, pp. 85–86, IEEE, 2018.
- [25] P. Baldi, *Deep learning in science*. Cambridge University Press, 2021.
- [26] F. Scarselli and A. C. Tsoi, "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results," *Neural networks*, vol. 11, no. 1, pp. 15–37, 1998.
- [27] Y. Patel, "Neural network training and non-differentiable objective functions," *arXiv preprint arXiv:2305.02024*, 2023.
- [28] R. DeVore, B. Hanin, and G. Petrova, "Neural network approximation," *Acta Numerica*, vol. 30, p. 327–444, 2021.
- [29] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753, PMLR, 09–15 Jun 2019.
- [30] L. Hertel, J. Collado, P. Sadowski, J. Ott, and P. Baldi, "Sherpa: Robust hyperparameter optimization for machine learning," *SoftwareX*, 2020. In press.
- [31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.
- [32] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Workshop on Machine Learning Systems*, 2017.
- [33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.