UC Irvine
UC Irvine Previously Published Works

Title
Sparse Phased Array Optimization Using Deep Learning

Permalink
https://escholarship.org/uc/item/68722864

Journal
2025 IEEE-APS TOPICAL CONFERENCE ON ANTENNAS AND PROPAGATION IN WIRELESS COMMUNICATIONS, APWC, 00(2025)

ISSN
2835-2378

ISBN
979-8-3315-4475-1

Authors
Lu, David Lin Yi
Maman, Lior
Earls, Jackson
et al.

Publication Date
2025-01-12

DOI
10.1109/apwc65665.2025.11190407

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/

eScholarship.org Powered by the California Digital Library
University of California

Sparse Phased Array Optimization Using Deep Learning

David Lin Yi Lu*, Lior Maman†, Jackson Earls*, Amir Boag†, Pierre Baldi*
* Department of Computer Science, University of California, Irvine, Irvine, USA
† School of Electrical and Computer Engineering, Tel Aviv University, Tel Aviv, Isreal
(lud19@uci.edu, maman@mail.tau.ac.il, jearls@uci.edu, boag@tauex.tau.ac.il, pfbaldi@ics.uci.edu)

***Abstract*—Antenna arrays are widely used in wireless communication, radar systems, radio astronomy, and military defense to enhance signal strength, directivity, and interference suppression. We introduce a deep learning-based optimization approach that enhances the design of sparse phased arrays by reducing grating lobes. This approach begins by generating sparse array configurations to address the non-convex challenges and extensive degrees of freedom inherent in array design. We use neural networks to approximate the non-convex cost function that estimates the energy ratio between the main and side lobes. This differentiable approximation facilitates cost function minimization through gradient descent, optimizing the antenna elements' coordinates and leading to an improved layout. Additionally, we incorporate a tailored penalty mechanism that includes various physical and design constraints into the optimization process, enhancing its robustness and practical applicability. We demonstrate the effectiveness of our method by applying it to the ten array configurations with the lowest initial costs, achieving further cost reductions ranging from 411% to 643%, with an impressive average improvement of 552%. By significantly reducing side lobe levels in antenna arrays, this breakthrough paves the way for ultra-precise beamforming, enhanced interference mitigation, and next-generation wireless and radar systems with unprecedented efficiency and clarity. 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 utilizing sparse configurations frequently encounter grating lobes, primarily due to limitations on the element count and beamwidth specifications that determine the antenna aperture size. These lobes often result from the periodic arrangements typical in conventional antenna arrays [1]. To address this, sparse arrays adopt aperiodic layouts. Nonetheless, designing these aperiodic arrays introduces significant challenges, such as an increased number of degrees of freedom to optimize and the substantial computational effort required to evaluate antenna radiation patterns across all main beam scan angles. Traditional techniques struggle to consistently identify global optima due to the complexity of the problem's landscape. The grid search method, though comprehensive, becomes impractical due to the myriad of possible configurations and the extensive computational resources needed.

Various strategies have been proposed to optimize thinned and sub-array configurations, employing advanced techniques to refine antenna design [2]–[8]. Additionally, prior methods focusing on sparse arrays have leveraged stochastic optimization techniques like simulated annealing to design aperiodic linear sparse arrays, aiming to minimize side lobe levels and optimize array configurations [9]. While these methods are adaptable and capable of handling complex, non-linear optimization challenges, they tend to be computationally demanding and slow, particularly for large arrays.

In recent years, machine learning techniques, particularly neural networks, have been increasingly applied to optimize complex issues in electromagnetics (EM). These networks effectively model computationally intensive EM simulations [10], [11] and intricate EM relationships [12], [13] with high accuracy. Beamforming, a signal processing technique, has been a focus of numerous deep learning applications [14]–[17]. Rather than inputting a desired array pattern into the neural network [17], our model takes a given sparse array as input and iteratively refines its geometry through gradient descent on the array coordinates.

To tackle computational speed and complexity, we initially reduce the number of variables in the phased array scanning problem from five (the observation vector, the scanning vector, and the wave number) to two [18]–[21]. Subsequently, we generate sparse array data using a sub-array method [22], which lessens the degrees of freedom in the optimization process. From this data, we select promising array configurations and further refine them 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 section, we tackle the challenge of designing Active Electronically Scanned Arrays (AESA) by simplifying the AESA array factor (AF) characterization from five variables to two and outlining the data generation process for optimization. The array factor (AF) for a planar phased array antenna, with uniformly excited elements situated in the yz-plane, is defined as follows:

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

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, $\hat{\mathbf{r}}_0 = \cos(EL_0)\sin(AZ_0)\hat{\mathbf{y}} + \sin(EL_0)\hat{\mathbf{z}} + \cos(EL_0)\cos(AZ_0)\hat{\mathbf{x}}$ indicates the main beam direction. Additionally, EL and AZ refer to the elevation and azimuth of the direction of observation, respectively, while EL_0 and AZ_0 relate to the main beam direction. It is noted that $EL = \pi/2 - \theta$ and $AZ = \varphi$ are in the conventional spherical coordinates system. The $\mathbf{r}_n = y_n\hat{\mathbf{y}} + z_n\hat{\mathbf{z}}$ specifies the location of the nth element and N_e represents the number of elements. By introducing a difference vector $\mathbf{u} = k(\hat{\mathbf{r}} - \hat{\mathbf{r}}_0)$, we redefine the array factor using only two variables:

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

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 based on 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 required for storage and processing, facilitating easier characterization of the entire observation area across the complete scan sector in a single illustration.

To streamline the optimization process and manage the search domain effectively, we adopted the concept of local periodicity to generate data [22]. This principle aids in organizing and integrating radiating elements and their associated transmit/receive modules through a sub-array strategy. In this approach, the entire array is constructed from multiple sub-arrays, each designed to exhibit periodicity. This strategy effectively narrows down the design variables. Practically, the array is configured with periodic sub-arrays, each capable of having distinct periodicity along both axes. These sub-arrays are also designed with the flexibility to be rotated and repositioned within their designated areas. The array's surface is segmented into various subdomains, each hosting a periodic sub-array. The primary design parameters for optimization include the periodicity along each axis, as well as the rotation and positional adjustments of each sub-array. This approach simplifies the complex task of array design by reducing the multitude of potential configurations to a structured set of variables. Fig. 1 illustrates an array configuration utilizing a periodic sub-array structure.

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

$$f_{cost} = -\frac{\sum_{i \in ML} |AF(u_y(i), u_z(i))|^{2p}}{\sum_{i \notin ML} |AF(u_y(i), u_z(i))|^{2p}} \tag{3}$$

where p determines the norm type and i is an index of samples in the (u_y, u_z)-plane. For our data generation, we set $p = 4$ to impose higher penalties on significant peaks, ensuring that excessively high sidelobes, such as grating lobes, do not appear outside the main lobe area.

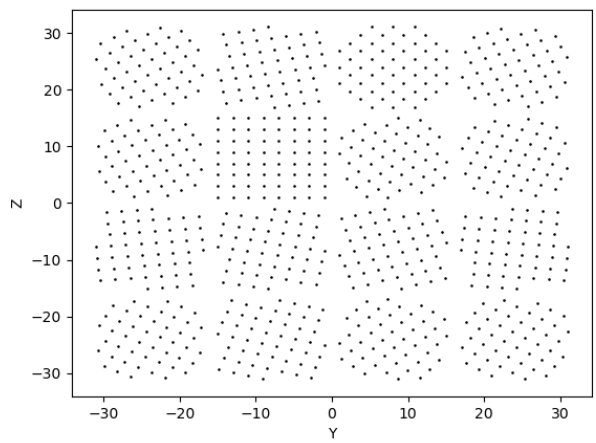

Fig. 1. Example of a periodic sub-array configuration.

III. Optimization of Antenna Arrays using Neural Network Approximations

Optimizing antenna array configurations is a formidable task, especially when aiming to minimize a complex and potentially non-differentiable cost function. The intricacy of the actual cost function demands considerable engineering efforts for direct optimization. To bypass these hurdles, we utilize neural networks as a strategic tool to accurately approximate the true cost function, acting as a surrogate. Neural networks are recognized for their universal approximation capability [23]–[26]. By nature, they provide a smooth and continuous function, which facilitates the use of gradient descent methods for optimization. The surrogate function is highly adaptable, as long as it provides an accurate estimate of the original cost function. This surrogate can adopt various forms, including feedforward neural networks (FNNs), convolutional neural networks (CNNs), or transformer variants, with the primary optimization process initiating after obtaining a reliable estimate. In our experiments, both an FNN and a Set Transformer [27] were evaluated as surrogate functions.

A. Model Selection

For the selection of optimal model architecture and hyperparameters, we utilize SHERPA, a hyperparameter optimization tool [28]. We conduct 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 FNN model architecture consists of four fully connected layers. The initial input layer is configured with 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. ReLU activation functions are applied after every fully connected layer to introduce non-linearity into the network. The final output layer is equipped with a single neuron, tasked with predicting the cost associated with the specified antenna array geometry.

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

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

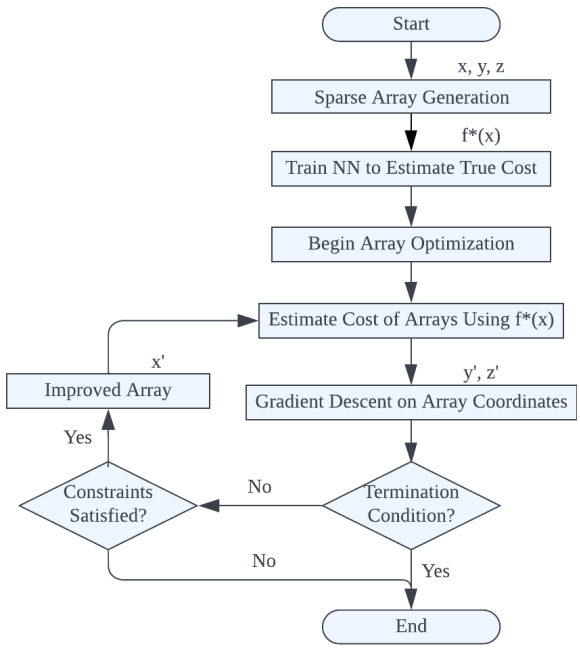

Fig. 2. Flowchart illustrating iterative optimization process

2) Set Transformer: The Set Transformer is designed to handle set-structured data, as outlined by Lee et al. [27]. It features an encoder and a decoder configuration. The encoder is equipped with two permutation equivariant set attention blocks to maintain order invariance of the inputs, while the decoder incorporates a multi-head attention block for pooling, two additional set-attention blocks, and a fully connected output layer.

Both the encoder and decoder are structured with attention blocks that each have two heads, and the hidden layers within sustain a dimensionality of 32. The model processes inputs with a dimension of 2, corresponding to the Y and Z coordinates of each antenna element, and the output layer, akin to the FNN model, includes a single neuron that calculates the cost for the antenna array configuration.

The training of the Set Transformer was conducted over 1000 epochs with a learning rate of 1×10^{-3}, a batch size of 64, using the Adam optimizer.

We evaluated different Set Transformer configurations by varying the hidden layer dimensions from 8 to 64 and the number of attention heads from 2 to 8. We explore learning rates between 1×10^{-2} and 1×10^{-6} and adjust batch sizes from 16 to 256 to determine optimal hyperparameters.

B. Cost

To evaluate the cost of an antenna array configuration, the neural network produces a real number, which spans from -40,000 to -0.7, with a lower cost signifying improved performance. Due to the broad spectrum of output values, we normalized these costs for training purposes and then utilized equation (4) to scale the training data. For the sake of interpretability, we apply reverse scaling to the outputs to restore the original cost values.

$$Y_{\text{scaled}} = \frac{Y - \mu_y}{\sigma_y} \tag{4}$$

$$Y = (Y_{\text{scaled}} \cdot \sigma_y) + \mu_y \tag{5}$$

Considering that our dataset was created using consistent parameters (same wavelength and antenna aperture size), the configuration cost is determined by the coordinates of the array elements. Each array is depicted as a set of coordinate pairs, featuring varying numbers of elements.

C. Inputs

The primary challenge in handling inputs arises from the neural network architecture selected. For architectures like the FNN used in our study, which are sensitive to the order and length of inputs, we address this issue by consistently ordering the antenna coordinates across all arrays. Additionally, to ensure uniform input length, we pad each input to match the maximum number of antennas in the dataset (1024).

Once the neural network provides a reliable estimate of the cost function, it forms the basis for optimizing antenna configurations. By structuring the configurations in a standardized input format, we enable the application of gradient-based optimization methods to refine antenna positions.

To optimize a specific antenna array configuration, we adjust the coordinates of the elements to find positions that minimize the cost function. Utilizing the neural network surrogate within a PyTorch framework [29], [30], we employ the Adam optimizer, a gradient-based optimization technique known for its computational efficiency and adjustable hyperparameters [31].

To enhance the optimization process further, we incorporate mechanisms that enforce physical constraints among the array elements. The first method is a minimum distance check, where each candidate configuration is evaluated at each iteration to ensure that elements maintain a set minimum distance constraint. If the candidate configuration satisfies this constraint, it is accepted, and the optimization advances to the next iteration. If it does not, the configuration reverts to the last valid arrangement, thereby concluding the optimization.

The second method involves adding 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 proceed uninterrupted. The penalty is designed to escalate significantly as any pair of elements nears the minimum distance threshold, actively discouraging configurations that breach spacing requirements. Designed to asymptotically approach infinity as the distance approaches the limit, this penalty can also be tailored to deter other

undesirable behaviors, such as elements spreading too far apart or converging too closely, ensuring that all configurations remain within acceptable limits throughout the optimization process.

D. Penalty Mechanism

In our model, the penalty mechanism is flexible, allowing for the incorporation of custom penalties tailored to specific design requirements. These penalties are designed to tackle various constraints or optimization goals. The following example demonstrates the penalty function we utilized.

This function computes the Euclidean pairwise distance between each pair of elements, denoted as D_{ij} for elements i and j, where i and j range from $1, 2, \ldots, N_e$. A minimum threshold is also established, represented as θ.

$$P_{ij} = \log\left(\frac{1}{D_{ij} - \theta}\right) \tag{6}$$

The total penalty P applied during the optimization is the aggregate of all individual penalties, scaled by a factor ϵ, which regulates the influence of the repulsion term on the optimization process:

$$P = \epsilon \cdot \sum_{\substack{\forall i \in \text{Ant} \\ \forall j \in \text{Ant}}} P_{ij} \tag{7}$$

Here, Ant denotes the set of elements in the configurations. The scalar ϵ is crucial as a higher value increases the emphasis on maintaining distance between elements, effectively preventing them from approaching the minimum threshold.

The objective of the optimization process is to minimize the custom loss function defined as

$$\text{loss} = f(\text{Ant}) + P \tag{8}$$

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

IV. Results

In this section, we showcase the performance improvements of the antenna arrays facilitated by the FNN and the Set Transformer, both with and without the implementation of the penalty mechanism designed to mitigate mutual coupling. Figure 3 displays the optimized sparse array generated by the FNN with the penalty mechanism incorporated. Figure 4 demonstrates the results for both the FNN and Set Transformer without the penalty mechanism, while Figure 5 shows the outcomes when the FNN is applied with and without the penalty mechanism. For simplicity, we chose the ten antenna array configurations with the lowest initial costs, with each optimization process taking approximately a minute to complete. Our method is readily scalable by executing multiple optimization processes in parallel.

The results indicate that the half-power beamwidth remains consistently similar ($\zeta_{3\text{dB}} = 0.78°$, where ζ represents either EL or AZ) across all scenarios, including the FNN with and without the penalty mechanism, and the Set Transformer. In

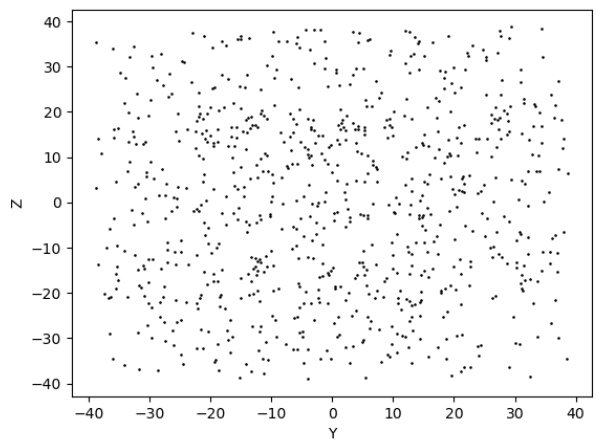

Fig. 3. Example of an optimized array configuration.

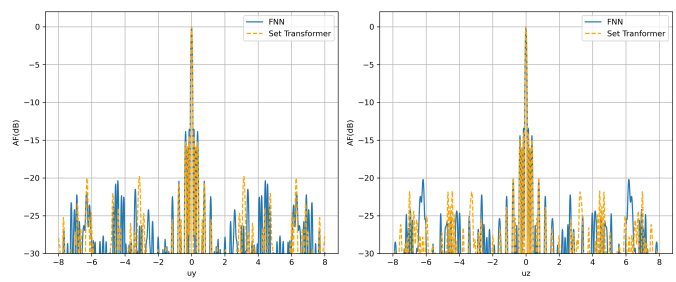

Fig. 4. The u_y and u_z cuts are derived using both the FNN and the Set Transformer. Solid lines indicate the results from the FNN, whereas dashed lines represent those obtained with the Set Transformer.

assessing the Side Lobe Level (SLL), we examine both the first and second SLL for a detailed comparison: the FNN with the penalty mechanism demonstrates the best performance, with the first SLL at (-17.46, -15.72) dB and the second SLL at (-20.03, -17.22) dB. Conversely, the Set Transformer and FNN without the penalty mechanism exhibit the first SLL at (-14.42, -14.06) dB and (-13.54, -13.42) dB respectively, and the second SLL at (-15.27, -14.93) dB and (-13.86, -14.41) dB, where the pair of values in parentheses represents the SLL measurements in u_y and u_z directions, respectively.

The FNN architecture incorporating the penalty mechanism demonstrates improved performance. A key approach to mit-

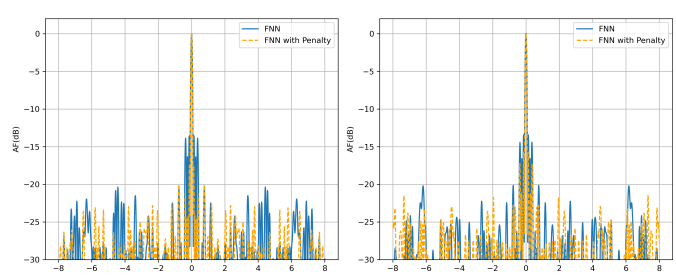

Fig. 5. The u_y and u_z cuts are derived using the FNN, both with and without the penalty mechanism. Solid lines indicate the results from the FNN without penalty, whereas dashed lines represent those obtained with penalty.

igating mutual coupling is to increase the minimum spacing between adjacent elements. With the penalty mechanism applied, the average minimum distance increases from 0.501λ (in the absence of the penalty) to 0.508λ.

TABLE I
COMPARISON OF COST BY OPTIMIZATION WITHOUT PENALTY

		FNN		SetTrans.	
Config.	Cost Bef.	Cost Aft.	%Chg	Cost Aft.	%Chg
0	-40463.98	-65314.69	-61.41%	-75943.74	-87.68%
1	-39931.30	-63976.30	-60.21%	-71023.64	-77.86%
2	-39767.27	-64469.17	-62.11%	-68749.44	-72.87%
3	-39472.41	-62825.44	-59.16%	-64694.44	-63.89%
4	-39431.64	-65592.67	-66.34%	-65318.07	-65.64%
5	-39383.22	-60226.57	-52.92%	-62768.85	-59.37%
6	-39036.94	-60053.17	-53.83%	-68111.58	-74.47%
7	-38702.13	-59404.24	-53.49%	-60284.31	-55.76%
8	-38690.57	-64205.93	-65.94%	-58751.75	-51.85%
9	-38600.70	-59656.36	-54.54%	-58166.98	-50.68%

A. Impact of Gradient Descent without Penalty

An analysis of the 10 lowest cost configurations, conducted without the use of the custom penalty, revealed an average cost reduction of 59%, with the most significant improvement observed at 66%. A closer examination of the minimum distances between elements after optimization shows that these distances often closely approached the threshold ($\theta = 0.5$). This proximity to the threshold indicates that the optimization process aims to minimize costs, but it may compromise 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, incorporating a penalty for the minimum distance constraint, has shown substantial improvements in the cost of antenna array configurations. With our custom penalty mechanism, we observed an average improvement of 552% across the configurations examined, with the largest improvement being 643% for the FNN model. These results highlight the potential of gradient descent to effectively refine antenna array designs to achieve significant performance enhancements.

The use of a penalty function in optimization demonstrates its ability to guide the optimization process within the bounds of operational feasibility while still maximizing cost reduction. Optimization with a penalty ensures that improvements in antenna array configurations do not compromise practical applicability. Moreover, 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 ϵ values of 12.5 for the FNN architecture and 1 for the Set Transformer architecture proved to be most optimal. These ϵ values were determined by testing ϵ in the range of 0.1 to 100.

TABLE II
COMPARISON OF COST BY OPTIMIZATION WITH PENALTY

		FNN		SetTrans.	
Config.	Cost Bef.	Cost Aft.	%Chg	Cost Aft.	%Chg
0	-40463.98	-207003.25	-411.57%	-125854.84	-211.02%
1	-39931.30	-253144.00	-533.94%	-74192.67	-85.80%
2	-39767.27	-254191.15	-539.19%	-82944.35	-108.57%
3	-39472.41	-240652.01	-509.67%	-82868.45	-109.94%
4	-39431.64	-265697.50	-573.81%	-76439.64	-93.85%
5	-39383.22	-265829.96	-574.98%	-81279.11	-106.38%
6	-39036.94	-282136.09	-622.74%	-92020.96	-135.72%
7	-38702.13	-234333.65	-505.47%	-83465.70	-115.66%
8	-38690.57	-287802.15	-643.85%	-77825.16	-101.14%
9	-38600.70	-273333.68	-608.10%	-66291.16	-71.73%

C. Comparative Analysis

When evaluating the results of optimization methods, the significance of our surrogate cost function becomes evident, as well as the correlation between cost reduction and the minimal distance among components.

Optimization using a penalty function not only ensured a safer margin from critical thresholds but also surpassed the performance of optimization without such penalties. This enhancement was noticeable with both the FNN and Set Transformer models. This approach guarantees that configurations stay within safe operational boundaries, thus improving the practical effectiveness and efficiency of the optimized antenna arrays.

This comparative study highlights the necessity of adhering to key design constraints. Incorporating a specific penalty term within the optimization process was markedly more successful than methods lacking this feature. Using this tailored penalty term led to more substantial cost reduction and increased separation between array elements.

V. CONCLUSION

We have introduced a new design methodology for antenna arrays that utilizes deep learning to efficiently optimize large antenna configurations. This approach begins with designing sub-arrays as the fundamental building blocks for configuring the entire array. At the core of our method is the application of neural networks to drive subsequent refinements in array geometry. Using a deep learning approach, we have demonstrated an efficient way to optimize large antenna arrays, significantly reducing costs while adhering to physical constraints. Specifically, we reduced the costs of the ten antenna array configurations with the lowest initial cost by an average of 552% with penalty and 59% without penalty. Our methodology is general and can be applied to other arrays, including non-planar arrays. Researchers can use this novel approach in future work for applications such as phase tapering, beam shaping, 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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] R. L. Haupt, *Antenna arrays: a computational approach*. John Wiley & Sons, 2010.

[7] R. L. Haupt, "Thinned arrays using genetic algorithms," *IEEE Transactions on Antennas and Propagation*, vol. 42, no. 7, pp. 993–999, 1994.

[8] R. L. Haupt, "Interleaved thinned linear arrays," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 9, pp. 2858–2864, 2005.

[9] A. Trucco and V. Murino, "Stochastic optimization of linear sparse arrays," *IEEE Journal of Oceanic Engineering*, vol. 24, no. 3, pp. 291–299, 1999.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] T. Lin and Y. Zhu, "Beamforming design for large-scale antenna arrays using deep learning," *IEEE Wireless Communications Letters*, vol. 9, p. 103–107, Jan. 2020.

[16] S. Komeylian, "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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] P. Baldi, *Deep learning in science*. Cambridge University Press, 2021.

[24] 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.

[25] Y. Patel, "Neural network training and non-differentiable objective functions," *arXiv preprint arXiv:2305.02024*, 2023.

[26] R. DeVore, B. Hanin, and G. Petrova, "Neural network approximation," *Acta Numerica*, vol. 30, p. 327–444, 2021.

[27] 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.

[28] L. Hertel, J. Collado, P. Sadowski, J. Ott, and P. Baldi, "Sherpa: Robust hyperparameter optimization for machine learning," *SoftwareX*, 2020. In press.

[29] 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.

[30] 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.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.