Adaptive Beamforming Algorithm Based on Automatic Deep Neural Network Optimization for Multiple Noise Signals

Zheng Xu[1,2], Zihao Pan[1], Daoxing Guo[1]*

[1] The College of Communications Engineering, Army Engineering University of PLA, China
[2] Nanjing Panda Handa Technology Co., Ltd., China
xuzhengjust@126.com, pzh199801@126.com, xyzgfg@sina.com

Abstract

As modern communication systems demand increasingly higher speed and accuracy in signal processing, traditional adaptive beamforming algorithms face challenges in real-time response to rapidly changing, multiple-noise signal environments. To address this issue, this paper proposes an Automated Deep Neural Network Adaptive Beamforming (A-DNNABF) algorithm for multiple noise signal environments. A-DNNABF uses the angle of arrival vector as input, employs an attention mechanism, and optimizes network architecture via Differentiable Architecture Search. Simulation results show A-DNNABF outperforms traditional Minimum Variance Distortionless Response (MVDR) and Deep Neural Network Adaptive Beamforming (DNNABF) methods in computational efficiency (10 times faster), prediction accuracy, and robustness to varying interference sources. The algorithm maintains stable performance with changing numbers of interference signals, demonstrating lower angular deviation in estimating both desired and interference signals. A-DNNABF provides an efficient solution for real-time adaptive beamforming in rapidly changing, multiple-noise signal environments.

Keywords: Adaptive beamforming, Hyperparameter optimization, Deep learning, Multiple noise signals

1 Introduction

Beamforming technology, as a core branch of array signal processing, plays a crucial role in various fields such as radar, sonar, communications, and electronic warfare. Traditional beamforming methods, such as algorithms based on the Signal-to-Noise Ratio (SNR) criterion, Minimum Mean Squared Error (MMSE) criterion, and Minimum Variance Distortionless Response (MVDR) criterion, require recalculating the weight vector when there are changes in the angle of the desired signal or interference signals. This process is computationally intensive and poses challenges for meeting the real-time systems' demand for quick updates of the optimal weight vector. Particularly when dealing with multiple noise signals, deep neural networks also struggle to respond promptly to adaptive beamforming.

With the rapid development of artificial intelligence and neural network technologies, applying them to the field of beamforming has become a new research direction. This approach utilizes neural networks for data pre-training and then employs the trained network model to adaptively output weight vectors during actual application phases. When the Angle of Arrival (AoA) of incoming signals changes, the neural network method processes with a minimal response time compared to traditional beamforming algorithms, which require extensive computation. This enables array antennas to achieve fast and efficient beamforming using neural network technology, significantly enhancing the system's realtime performance. However, the angle of arrival of both the desired signal received by the array antenna and the multiple interference signals often change rapidly. Traditional beamforming algorithms based on deep neural networks struggle to respond in real-time and provide optimal weight vector matrices due to the rapid variation of input noise sources. To address this issue, this paper proposes an Automated Deep Neural Network Adaptive Beamforming for Multiple Noise Signal (A-DNNABF) algorithm. This algorithm takes the AOA vector of the incoming signal as network input, enhances the learning of key features through an attention mechanism, and automatically learns the architecture of a deep neural network from large-scale input signals to output weight vectors that approximate the MVDR algorithm. Simulation results demonstrate that, compared to MVDR and DNNABF, the A-DNNABF algorithm not only accurately fits the MVDR algorithm's weight vectors but also adapts more effectively to rapid changes in the AoA of incoming signals, adaptively forming beams and nulls. Moreover, the computational speed of the A-DNNABF algorithm is approximately 10 times faster than that of the MVDR algorithm.

Regarding the application scenario, our method primarily addresses signal processing challenges in complex electromagnetic environments. Specifically, it is applicable to high-density signal environments such as urban communication networks, large industrial facilities, or military communication systems. In these scenarios, issues like signal interference, multipath effects, and spectrum congestion are prevalent. Our approach captures real environmental characteristics through large-scale

*Corresponding Author: Daoxing Guo; Email: xyzgfg@sina.com
DOI: https://doi.org/10.70003/160792642025112606003

signal collection, adaptively processes complex signals using DARTS-optimized neural network architectures, and achieves real-time signal quality optimization through weight prediction. This makes our technology excel in improving communication quality, reducing signal conflicts, and enhancing anti-interference capabilities, particularly suitable for communication systems requiring high reliability and real-time performance.

The remainder of this paper is organized as follows. Section 2 briefly introduces related work. In Section 3, we detail the design of the proposed method A-DNNABF. Section 4 describes the experimental setup, and Section 5 presents our experimental results comparing the performance of A-DNNABF with some benchmarks. Finally, in Section 6, we summarize the paper and propose future work.

2 Related Work

2.1 Beamforming Algorithm Based on MVDR Beamformer

Consider a uniform linear array with M elements and an element spacing of d. Assume there are $K + 1$ incoming signals, where the angle of arrival of the desired signal is θ_0, and the angles of arrival of the K interference signals are θ_k (k = 1, 2, ..., K). The vector composed of the angles of arrival (AOA) of the incoming signals is $\theta = [\theta_0, \theta_1, ..., \theta_k]$. The array output signal can be expressed as:

$$x(t) = \sum_{k=0}^{K} s_k(t) a(\theta_k) + n(t) \tag{1}$$

where $s_k(t)$ is the complex envelope of the k-th signal, $a(\theta_k)$ is the corresponding array manifold vector, and $n(t)$ is the additive white noise vector.

The goal of the MVDR beamformer is to minimize the output power while maintaining a unit gain in the direction of the desired signal. The computation of its weight vector is as follows:

1) Array Output Signal Model:

$$x(t) = a(\theta_0) s_0(t) + \sum_{k=1}^{K} a(\theta_k) s_k(t) + n(t) \tag{2}$$

where: $x(t)$ is the $M \times 1$ array received signal vector, $a(\theta_0)$ is the $M \times 1$ array manifold vector in the direction of the desired signal, $s_0(t)$ is the complex envelope of the desired signal, $a(\theta_k)$ is the array manifold vector in the direction of the k-th interference signal, $s_k(t)$ is the complex envelope of the k-th interference signal, and $n(t)$ is an $M \times 1$ additive white noise vector.

2) Computing the Array Covariance Matrix:

$$R = E[X(t) X^H(t)] \tag{3}$$

Where $E[\cdot]$ denotes the expectation operation, and $(\cdot)^H$ represents the conjugate transpose. R is an $M \times M$ matrix containing spatial correlation information of signals, interference, and noise.

3) MVDR Optimization Problem: The weight vector w of the MVDR beamformer is obtained by solving the following optimization problem:

$$\min_w w^H R w \quad \text{s.t. } w^H a(\theta_0) = 1 \tag{4}$$

The goal of this optimization problem is to minimize the output power $w^H R w$ while maintaining a unit gain constraint in the direction of the desired signal, $w^H a(\theta_0) = 1$.

4) Solution to the MVDR Weight Vector: Using the Lagrange multiplier method, a closed-form solution for the MVDR weight vector can be obtained:

$$w_{MVDR} = \frac{R^{-1} a(\theta_0)}{a^H(\theta_0) R^{-1} a(\theta_0)} \tag{5}$$

Where: R^{-1} is the inverse of the array covariance matrix, and $a(\theta_0)$ is the array manifold vector in the direction of the desired signal.

5) Array Output: Using the MVDR weight vector, the array output can be expressed as:

$$y(t) = w_{MVDR}^H x(t) \tag{6}$$

This output maintains a unit gain response for the desired signal while minimizing the total output power, effectively suppressing interference and noise. The MVDR can adaptively adjust the weights based on the statistical properties of the received signal to optimally suppress interference and noise. Under high signal-to-noise ratio conditions, the MVDR can achieve higher spatial resolution compared to traditional beamforming methods. However, the MVDR requires the computation of the covariance matrix inverse, which may lead to high computational complexity when the number of array elements is large. To accurately estimate the covariance matrix, the MVDR typically requires a sufficient number of samples, which could be a challenge in rapidly changing environments.

2.2 Deep Neural Networks for Beamforming

Currently, numerous studies focus on neural network-based beamforming technologies. Hopfield networks, principal component analysis networks, and multilayer perceptrons have been utilized for beamforming; however, these methods do not fully exploit the nonlinear fitting capabilities of neural networks [1-2]. Radial-basis function (RBF) networks map the relationship between the received signal covariance matrix and the weight vector for beamforming, but clustering operations are necessary to obtain the centers of the basis functions. The training process of these networks is complicated, and as RBF

networks are single-hidden-layer feedforward networks, the number of neurons in the hidden layer increases significantly with more training samples, resulting in an overly large network structure [3].

To overcome the drawbacks of RBF networks, deep neural network (DNN) technology has begun to be applied to beamforming. [4-6] used convolutional neural networks (CNNs) to replace complex optimization algorithms, predicting approximate optimal weight vectors by inputting the autocorrelation matrix of received signals. However, this method necessitates cumbersome data preprocessing steps for better fitting results and a simpler network architecture, introducing excessive angle-related prior assumptions. Ren et al. [7] proposed a new algorithm based on existing research and identified issues called Deep Neural Network Adaptive Beamforming (DNNABF). It inputs vectors composed of random desired and interference signal angles of arrival (AOA) and outputs the MVDR algorithm weight vector. It uses four hidden layers activated by the PReLU function, with the output layer activated by the tanh function. It employs the Adam algorithm as the optimizer, granting the network strong generalization capabilities. This allows it to adaptively form beams and nulls at the desired and interference signal angles when the angles of arrival change, performing real-time anti-interference in the spatial domain and accommodating more complex interference scenarios.

2.3 Hyperparameter Optimization Techniques

In deep learning, hyperparameter optimization (HPO) is finding optimal combinations of hyperparameters to enhance model performance. Since hyperparameters directly affect the model's learning process and final performance, correctly selecting and adjusting them is crucial. The working principle of HPO algorithms is consistent: they iteratively propose hyperparameter configurations and then evaluate their performance. These hyperparameters and their respective evaluations are continuously stored in what is termed an archive, and if the model achieves optimal performance, the desired hyperparameter configuration is obtained [8]. Below are some commonly used hyperparameter optimization methods:

2.3.1 Grid Search (GS)

Grid search involves discretizing the range of each hyperparameter (HP) and exhaustively evaluating every combination of values. Numerical and integer HP values are typically evenly distributed within their fixed constraints. The number of different values for each HP is referred to as the grid's *resolution*. For categorical HPs, part or all possible values are considered. This is an exhaustive search method that systematically explores multiple hyperparameter combinations within a predefined space. It is suitable when the number of hyperparameters is small, and the range of potential values for each hyperparameter is limited. GS is directly affected by the curse of dimensionality [9] since the required number of evaluations grows exponentially with the number of HPs when the grid resolution is fixed.

2.3.2 Random Search (RS)

Unlike grid search, random search selects hyperparameter combinations within the possible range. This method is often faster than grid search, especially when the hyperparameter space is large. In its simplest form, each hyperparameter value is sampled independently from a pre-specified (usually uniform) distribution, applicable to numeric, integer, or categorical parameters with box constraints. RS performs better than GS in high-dimensional HPO settings [10].

2.3.3 Bayesian Optimization

Bayesian optimization uses Bayesian statistics to select hyperparameters to optimize the objective function (typically the model's performance on the validation set). It builds a probabilistic model between hyperparameters and the objective function and selects hyperparameters that are most likely to improve performance. Bayesian Optimization (BO), as a global optimization technique for black-box functions, especially for HPO [11-13], has become increasingly popular.

2.3.4 Evolutionary Algorithms

Evolutionary Strategies (ES) are a class of stochastic population-based optimization methods, inspired by the concept of biological evolution, falling under the broader category of Evolutionary Algorithms. In ES terminology, an individual is a single HPC, a population is the currently maintained set of HPCs, and the fitness of an individual is its (inverted) generalization error. Mutation is a (random) change of one or several hyperparameter values in a configuration. Crossover creates a new HPC by (randomly) combining values from two other configurations. An ES follows iterative steps to find individuals with high fitness values. Using so-called nested ES, ES are less likely to get stuck in local minima [14], and they can be directly modified to improve noise resilience [15]. Furthermore, ES can be applied to settings with complex search spaces and thus can work in spaces where other optimizers might fail [16]. ES is more efficient than RS and GS but still typically requires many iterations to find good solutions, making them perform poorly in expensive optimization settings like HPO.

2.3.5 Gradient-Based Methods

Gradient-based hyperparameter optimization techniques use gradient information to optimize hyperparameters, mainly for differentiable hyperparameters. The core idea of this approach is to directly optimize hyperparameters by computing their gradients to improve model performance. This method requires hyperparameters to be differentiable concerning the loss function; compared to random search and evolutionary algorithms, using gradient information can find optimal solutions more quickly. It can continually adjust hyperparameters during training. Differentiable Architecture Search (DARTS) [17] defines a continuous search space and optimizes architecture parameters using gradient descent. This method is increasingly used in deep learning, particularly in automated machine learning (AutoML).

When performing hyperparameter optimization, the introduction of hyperparameter optimization libraries such

as Hyper-opt, Optuna, Ray Tune, and Scikit-optimize [18-21] can facilitate rapid hyperparameter optimization. These libraries provide tools for executing the aforementioned methods and sometimes include more advanced features. Factors to consider during the optimization process include:

- Evaluation Metrics: Clearly define evaluation metrics for model performance. This will determine the objective of the optimization process.
- Validation Strategy: Use methods like cross-validation to assess the performance of hyperparameter combinations.
- Computational Resources and Time: Hyperparameter optimization is typically computation-intensive. Plan the allocation of computational resources and time reasonably.
- Overfitting Avoidance: Pay attention to performance on the validation set to avoid choosing hyperparameter combinations that lead to overfitting.

Hyperparameter optimization is an iterative process requiring multiple experiments to find the optimal combination. In practice, experience and intuition also play important roles.

3 Method

Our method consists of three core components: Large-Scale Signal Collection, Signal Processing Neural Network Architecture Optimization Based on DARTS, and Weight Prediction Output. First, we collect and preprocess large-scale signal data. Then, this data is input into a DARTS-based optimization module to automatically search for the optimal neural network architecture. Finally, the optimized network is used for weight prediction, outputting optimal processing weights based on input signal characteristics. These three components form a closed-loop system, with data flowing sequentially between stages while allowing feedback optimization, continuously improving the efficiency and accuracy of signal processing.

The motivation behind our proposed new signal processing method stems from several key challenges faced by current technologies. Firstly, existing beamforming methods based on convolutional neural networks are ineffective in handling dynamically changing interference signal sources. Secondly, while the MVDR algorithm is widely used in signal processing, its computational efficiency is relatively low, making it difficult to meet real-time processing requirements. Additionally, the rapid development of deep neural networks, especially the significant advantages demonstrated by Transformer models based on attention mechanisms in capturing temporal signals, has provided us with new research directions.

3.1 Large-Scale Signal Collection

Algorithm 1 describes a large-scale signal collection process aimed at providing training data for subsequent neural network architecture optimization. The algorithm first defines the basic parameters of an array antenna, modeled as a linear array, including the number of linear antenna elements M and spacing d, as well as signal sampling parameters such as sampling frequency f_s and sampling time T.

Algorithm 1. A large-scale signal collection process

1: Define array antenna parameters: number of array elements M, spacing between elements d
2: Define sampling parameters: sampling frequency f_s, sampling time T
3: Initialize signal set $S = \{\}$
4: **for** each scenario $i = 1$ to N **do**
5: Generate expected signal AOA $\theta_0^{(i)}$ and K interference signal AOAs $\left\{\theta_k^{(i)}\right\}_{k=1}^{K}$
6: Construct AOA vector $\theta^{(i)} = \left[\theta_0^{(i)}, \theta_1^{(i)}, \cdots, \theta_K^{(i)}\right]$
7: Generate corresponding signal $x^{(i)}(t)$, $t \in [0, T]$
8: Compute MVDR weight vector $w_{MVDR}^{(i)}$
9: $S = S \cup \left\{\left(\theta^{(i)}, x^{(i)}(t), w_{MVDR}^{(i)}\right)\right\}$
10: **end for**
11: **return** signal set S

In the main loop, the algorithm simulates N different signal scenarios. For each scenario i, it generates an expected signal angle of arrival (AOA) $\theta_0^{(i)}$ and K interference signal AOAs $\left\{\theta_K^{(i)}\right\}_{k=1}^{K}$. These AOA values are combined into a vector $\theta^{(i)}$, representing the spatial signal distribution of the current scenario.

Based on these AOAs, the algorithm generates the corresponding time-domain signal $x^{(i)}(t)$, $t \in [0, T]$. This signal contains a mix of the expected signal and interference, simulating a real complex signal environment. Simultaneously, the algorithm computes the minimum variance distortionless response (MVDR) weight vector $w_{MVDR}^{(i)}$ for the scenario.

Finally, the algorithm adds the AOA vector $\theta^{(i)}$, the time-domain signal $x^{(i)}(t)$, and the MVDR weight vector $w_{MVDR}^{(i)}$ as a whole to the signal set S. This process is repeated N times, ultimately generating a signal set containing a large number of different scenarios.

The signal set S will serve as input for subsequent neural network architecture optimization algorithms, used to train and evaluate the network's performance in different signal processing scenarios.

3.2 Signal Processing Neural Network Architecture Optimization Based on DARTS

We will use the Transformer based on the attention mechanism as the basic network architecture and incorporate Neural Architecture Search (NAS) technology to automatically optimize the network structure, primarily designing an automated deep neural network architecture optimization algorithm using Differentiable Architecture Search (DARTS).

Compared to Progressive Neural Architecture Search (PNAS), DARTS is generally faster because it does not need to gradually expand the search space. DARTS might be easier to implement and adjust because it employs standard gradient descent methods; compared to Efficient Neural Architecture Search (ENAS), DARTS is usually easier to implement because it does not require a complex reinforcement learning controller. The search process in DARTS may be more stable because it is based on gradient methods rather than sampling methods. The hyperparameter optimization process, based on DARTS, effectively explores the entire hyperparameter landscape. This search process implicitly evaluates the algorithm's sensitivity to different hyperparameter combinations. The optimal hyperparameters reported in our study represent the best configuration found after exploring a wide range of possibilities, ensuring both robustness and high performance.

3.2.1 Problem Formulation

Given the input signal set $S=\{(\theta^{(i)},x^{(i)}(t),w^{(i)}_{MVDR})\}_{i=1}^{N}$, where $\theta^{(i)}$ is the angle of arrival (AOA) vector, $x^{(i)}(t)$ is the time-domain signal, and $w^{(i)}_{MVDR}$ is the MVDR weight vector. Our goal is to find an optimal neural network architecture $\mathcal{A}^*$ such that:

$$\mathcal{A}^* = \arg\min_{\mathcal{A}} \mathcal{L}_{\text{val}}(\mathcal{A}, w^*(\mathcal{A})) \tag{7}$$

where $w^*(\mathcal{A})$ represents the optimal network weights for a given architecture $\mathcal{A}$, and $\mathcal{L}_{\text{val}}$ is the loss function on the validation set.

3.2.2 Search Space Definition

This method was proposed in the original Transformer paper. It uses sine and cosine functions of different frequencies to encode positional information. (1) Learnable Position Encoding: This method treats positional encoding as learnable parameters. The model can adaptively learn position representations best suited for the task during training. (2) Relative Position Encoding: Unlike absolute position encoding, this method encodes the relative distances between elements. This is particularly useful when dealing with variable-length sequences and can help the model better capture local structures. (3) Rotary Position Encoding (RoPE): This is a newer method that encodes relative positional information through complex number rotation. It theoretically offers infinite extrapolability and is more robust to variations in sequence length. (4) ALiBi (Attention with Linear Biases): This method introduces linear biases in attention computation to implicitly encode positional information. It does not require explicit position embeddings and is especially effective for long sequences. (5) Fourier Position Encoding: This utilizes the Fourier series to encode positional information, which can better handle periodic signals or data with multi-scale structures.

Skip Connections, also known as Residual Connections or Shortcut Connections, are a widely-used technique in deep neural networks. Skip connections allow the direct transmission of input from one layer to a deeper layer, bypassing some intermediate layers. This connection enables information to flow more freely through the network, not strictly following a hierarchical forward propagation path. They can be used to alleviate the vanishing gradient problem, promote feature reuse, and simplify the optimization process. In DARTS-based neural network architecture optimization for signal processing, skip connections can serve as an option in the search space. We can include various types of skip connections in the search space, such as simple identity mappings, skip connections with linear transformations, or those with lightweight convolutions. The algorithm can automatically decide where to add skip connections within the network. This may include adding skip connections between Transformer blocks, between self-attention layers and feedforward network layers, or long-range skip connections across multiple layers. By including skip connection options in the search space, our algorithm can automatically discover the network topology that is best suited for specific signal-processing tasks. This flexibility allows the algorithm to balance between different network depths and complexities while maintaining good gradient flow and feature propagation. In signal processing tasks, skip connections can be particularly useful because they help models better handle information across different time scales or frequency ranges while retaining direct access to the original input signal. This could be beneficial for accurate prediction of MVDR weight vectors and AOA estimation.

3.2.3 Mixed Operation Definition

A mixed operation is the weighted sum of all candidate operations in the search space. For each learnable connection in the network, we define a mixed operation that includes all possible candidate operations (such as multi-head self-attention, feedforward networks, etc.), with each operation having an associated weight parameter. In our signal processing task, mixed operations may include different types of attention mechanisms (such as multi-head self-attention with different numbers of heads), various configurations of feedforward networks, and various positional encoding methods and skip connection options. By optimizing the weights of these mixed operations, the algorithm can automatically discover the network structure that best handles the given signal data, thereby achieving superior performance in tasks like MVDR weight vector prediction.

For each learnable connection (i, j) in the network, we define the mixed operation as:

$$\overline{o}^{(i,j)}(\mathrm{x}) = \sum_{o\in\mathcal{O}} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o'\in\mathcal{O}}\exp\left(\alpha_{o'}^{(i,j)}\right)} o(\mathrm{x}) \tag{8}$$

Where $\alpha_o^{(i,j)}$ is a learnable architecture parameter.

3.2.4 Architecture Design

The final network architecture will consist of the following components:

- Input Layer: Processes the time-domain signal $x^{(i)}(t)$

- Feature Extraction Layer: Utilizes the attention mechanism obtained through search
- Transformer Encoder: Employs the optimal self-attention mechanism and feedforward network obtained through search
- Output Layer: Predicts the weight vector $\hat{w}$

3.2.5 Optimization Algorithm

We propose Algorithm 2 to optimize the neural network architecture: The loss function $\mathcal{L}$ can be defined as:

$$\mathcal{L} = MSE\ (\hat{w}, w_{MVDR}) \tag{9}$$

where $\hat{w}$ is the weight vector predicted by the network, and MSE stands for mean squared error.

Algorithm 2. DARTS-based signal processing neural network architecture optimization

1: Input: Signal set S, search space $\mathcal{O}$
2: Initialize architecture parameters α and network weights w
3: **while** convergence condition is not met **do**
4: Sample batch data $\mathcal{B}$train, $\mathcal{B}$val from S
5: Update weights: $w \leftarrow w - \eta_w \nabla_w \mathcal{L}$train $(w^*, \alpha; \mathcal{B}$train$)$
6: Update architecture: $\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha \mathcal{L}$val $(w^*(\alpha), \alpha; \mathcal{B}val)$
7: **end while**
8: Determine final architecture $\mathcal{A}$ based on α
9: **return** $\mathcal{A}$

Algorithm 2 first takes a set of signals and a predefined search space as input. The search space includes various possible network operations, such as different configurations of multihead self-attention mechanisms, feedforward network layers, position encoding methods, and skip connections. By introducing the concept of mixed operations, the algorithm transforms these discrete choices into a continuous optimization problem. During the initialization phase, the algorithm randomly sets the architecture parameters α and network weights w.

The algorithm then enters an iterative optimization process. In each iteration, it samples training and validation batch data from the input signal set. The optimization process consists of two alternating steps: first, the network weights w are updated using the training batch to minimize the training loss. Next, the architecture parameters α are updated using the validation batch to minimize validation loss. This bilevel optimization strategy allows the algorithm to simultaneously consider both network performance and structural efficiency. The loss function is defined as the mean squared error (MSE) between the predicted weight vector and the MVDR (Minimum Variance Distortionless Response) weight vector. This choice reflects the algorithm's specific application in signal processing tasks, namely, accurately predicting the MVDR weight vector. The iterative process continues until a predetermined convergence condition is met. Finally, based on the optimized architecture parameters α, the algorithm determines the final network architecture. This architecture is the most suitable structure within the given search space to execute the specific signal processing task.

3.3 Weight Prediction Output

In the final stage, we use the optimized Transformer model to predict the MVDR weight vector. The prediction process of the weight vector is shown in Algorithm 3.

Algorithm 3. Weight prediction output

1: Input: Optimized Transformer model M_{a^*}, AOA vector θ
2: Encode the AOA vector: $e = Encode(\theta)$
3: Process through the Transformer model: $h = M_{a^*}(e)$
4: Generate weight vector through the fully connected layer: $\hat{w} = FC(h)$
5: **return** Predicted weight vector $\hat{w}$

Algorithm 3 describes the process of using an optimized Transformer model to predict the MVDR weight vector. The algorithm receives two key inputs: a Transformer model M_{a^*} optimized through architecture search and a vector θ representing the angle of arrival (AOA) of the signal.

First, the algorithm encodes the input AOA vector: $e = Encode(\theta)$. This encoding process may include positional encoding or other forms of signal representation transformation, aiming to convert AOA information into a format that can be effectively processed by the Transformer model.

Next, the encoded vector is input into the optimized Transformer model: $h = M_{a^*}(e)$. The Transformer model processes this input through its self-attention mechanism and feedforward network layers, generating a high-dimensional feature representation h. This process allows the model to capture complex relationships and patterns between AOAs.

Finally, this high-dimensional feature is processed through a fully connected layer to output the predicted MVDR weight vector: $\hat{w} = FC(h)$. This weight vector $\hat{\mathbf{w}}$ represents the model's prediction of the optimal beamforming strategy for the given AOA.

The entire process achieves an end-to-end mapping from AOA information to the MVDR weight vector, fully leveraging the optimized Transformer architecture to capture complex patterns and relationships in signal processing.

Automated hyperparameter optimization theoretically enhances the performance of the A-DNNABF algorithm by systematically exploring a larger hyperparameter space than manual tuning. This approach leverages adaptive search strategies to efficiently navigate the complex relationship between hyperparameters and model performance. By optimizing multiple objectives simultaneously, such as accuracy and computational efficiency, it can find a better balance in the bias-variance trade-off. The automated process also has the potential to discover non-intuitive hyperparameter combinations that human experts

might overlook, leading to improved generalization and more robust performance across various operational conditions. Ultimately, this systematic approach increases the likelihood of finding an optimal or near-optimal configuration, resulting in superior algorithm performance compared to traditional manual tuning methods.

4 Experiments

4.1 Datasets

This subsection introduces the dataset used in the experiment. It is assumed that there is 1 desired signal and 8 interference signals incident as plane waves. The max 9 incident signals are taken from different angles and randomly assigned. A total of 600,000 sets of sample data are generated, among which 400,000 sets are used for training, 100,000 sets for model validation, and 100,000 sets for testing. This dataset is based on the MVDR algorithm and is calculated and solved according to formula 5. Our simulation-based approach effectively meets the testing requirements by this dataset for this study. Simulations allow for systematic testing under various controlled conditions, which is crucial for rigorously evaluating the A-DNNABF algorithm's performance across a wide range of scenarios. This controlled environment enables us to isolate specific variables, test extreme cases, and ensure reproducibility - aspects that can be challenging to achieve in real-world settings. Moreover, our simulations are based on well-established models that closely mimic real-world conditions, providing a reliable proxy for algorithm performance.

4.2 Comparison Baselines

In our experiment, we selected two state-of-the-art adaptive beamforming algorithms as our baselines to compare with our proposed A-DNNABF beamforming algorithm.

4.2.1 MVDR

We adopted the Minimum Variance Distortionless Response (MVDR) proposed by Capon [22] as a baseline algorithm to evaluate the computational efficiency of different algorithms. This algorithm is a classic adaptive beamforming algorithm used for processing signals received by a sensor array. Its goal is to maximize the gain of the received signal in a specific direction by adjusting the weights of the array.

4.2.2 DNNABF

Ren et al. [7] proposed an adaptive beamforming algorithm based on deep neural networks (Deep Neural Network Adaptive Beamforming, DNNABF), which uses a vector composed of the AOA of incident signals as network input. The network output approximates the weight vector obtained by the Minimum Variance Distortionless Response (MVDR) algorithm.

4.3 Experimental Environment

The simulations in this paper are conducted using an Intel(R) Core(TM) i7-10700K CPU @ 3.80GHz, with 96GB of memory, on the Pytorch 2.0 simulation platform. A uniform linear array is adopted, with simulation parameters set as follows: the number of array elements is 12, the spacing between elements is half a wavelength, the signal-to-noise ratio is 10dB, the interference-to-noise ratio is 30dB, the number of snapshots is 1024, the angle of arrival ranges from [-90°, 90°] degrees, with a step of 1°.

5 Experimental Results and Analysis

In this section, we address research questions related to beamforming based on deep neural networks through empirical studies. We focus on algorithm efficiency, weight matrix accuracy, and algorithm stability, discussing these issues by answering three research questions.

RQ1. Can our method achieve optimal computational efficiency compared to baseline weight matrix computation algorithms?

The primary motivation for exploring this question stems from the urgent need for computational efficiency in practical signal processing applications. Although the traditional MVDR algorithm can provide high-quality beamforming weight vectors, its computational complexity is high, which can become a bottleneck in scenarios requiring real-time processing of large amounts of signal data. Our method, based on an optimized Transformer architecture, aims to address this issue by leveraging the parallel computing capabilities and efficient feature extraction abilities of deep learning models, potentially reducing computation time while maintaining the accuracy of weight vector predictions.

To objectively evaluate the computational efficiency of our method, we use the average running time on a test dataset as the primary evaluation metric. This metric directly reflects the algorithm's response speed in practical applications, which is particularly important for signal systems requiring realtime processing. We compare our method with the traditional MVDR algorithm on the same test dataset and hardware environment to ensure fairness and comparability of the evaluation. The experimental results show that the traditional MVDR algorithm takes an average of 0.3158 seconds to compute a weight vector, whereas our method based on an optimized Transformer takes only 0.0325 seconds, achieving a nearly tenfold increase in computational speed. This remarkable performance improvement can be explained from the perspective of algorithm time complexity. The traditional MVDR algorithm involves a matrix inversion operation with a time complexity of $O(M)^3$, where M is the number of array elements. In contrast, our method primarily relies on the forward propagation of the Transformer, with time complexity linearly related to the input sequence length and highly parallelizable. Furthermore, after training, our method only requires simple matrix multiplication operations during the inference stage, further reducing computation time.

Our algorithm's key advantage lies in its ability to achieve superior execution efficiency under equivalent hardware conditions. This efficiency stems from the fact that the process of predicting weights is essentially a deep

neural network inference process, which is computationally less demanding than traditional adaptive beamforming methods.

Theoretically, the computational complexity of our algorithm is primarily determined by the forward pass of the neural network, which has a time complexity of O(n), where n is the number of parameters in the network. In contrast, traditional MVDR has a complexity of $O(M)^3$, where M is the number of array elements, due to the matrix inversion operation.

Moreover, the inference process in our algorithm can be efficiently executed on various hardware platforms, including CPUs. This is because deep learning frameworks have been highly optimized for inference tasks, leveraging techniques such as model quantization and efficient memory management. As a result, our method can achieve real-time performance even on resource-constrained devices.

Answer to RQ1: Our proposed method based on the optimized Transformer architecture significantly surpasses the traditional MVDR algorithm in terms of computational efficiency. This result not only validates the effectiveness of our method but also offers a promising solution for real-time signal processing systems. Particularly in scenarios that require handling large volumes of data or have very high demands on latency, our method may bring substantial performance improvements. However, it is noteworthy that while computational efficiency has greatly improved, we still need to further verify the accuracy and stability of the method in different signal environments to ensure its reliability in practical applications.

RQ2. Does our proposed algorithm achieve minimal prediction error?

The main motivation for investigating this question arises from the ongoing demand for higher accuracy in predictions within the field of adaptive beamforming. Although the traditional MVDR algorithm can provide the theoretically optimal solution, its performance may degrade in practical applications due to various factors. The adaptive beamforming algorithm based on deep neural networks (DNNABF) proposed by Ren et al. is a significant breakthrough, demonstrating the potential of deep learning in this area. However, we believe that by optimizing the network architecture and training strategies, further improvements in prediction accuracy can be achieved. Our method, based on an optimized Transformer architecture, aims to overcome the limitations of DNNABF and explore whether more accurate weight vector predictions can be achieved while maintaining high computational efficiency.

When the desired signal and eight interference signals reach the linear array, the antenna patterns obtained by the DNNABF and A-DNNABF methods are shown in Figure 1. It can be seen that both algorithms are capable of forming a beam at the angle of the desired signal and forming nulls at the angles of the interference signals. The angle of arrival for the desired signal is 18°, and the angles of arrival for the 8 interference signals are -32°, -20°, -14°, -4°, 7°, 25°, 36°, and 58°. The number of test datasets is 5000.

To objectively assess the prediction accuracy of our method, we utilized the mean square error (MSE) between actual values and network-predicted values as the primary evaluation metric. This metric directly reflects the accuracy of the algorithm's predictions and is crucial for evaluating the network's generalization capability. Lower MSE indicates better generalization performance, meaning that the algorithm can adapt better to unseen data and reduce the risk of overfitting. We evaluated our method on test datasets of different scales (with sample sizes of 5000, 10000, and 15000) to ensure the reliability and stability of the results. By comparing our method's performance with the DNNABF algorithm on the same test sets, we can comprehensively evaluate the prediction accuracy and generalization capability of both methods.

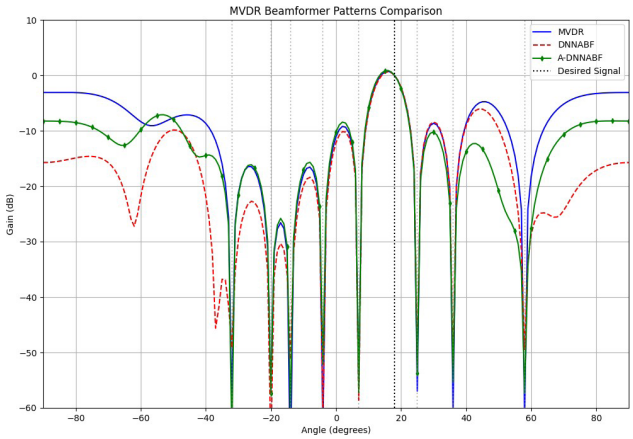


Figure 1. MVDR, DNNABF, and A-DNNABF antenna patterns

The experimental results show that our proposed method, based on the optimized Transformer architecture, significantly outperforms the DNNABF algorithm across all scales of test data, with test errors consistently remaining at a lower level. This result can be explained from the perspective of network architecture. Our method is based on the Transformer architecture, whose self-attention mechanism can effectively capture long-range dependencies between signals, which is particularly important when dealing with complex spatial signal distributions. In contrast, DNNABF mainly relies on traditional feedforward neural networks, which may have limitations in capturing complex interactions between signals. Additionally, our method has been optimized through architecture search, possibly finding a network structure more suitable for beamforming tasks. Finally, the positional encoding mechanism of the Transformer may better preserve the spatial information of the signals, further improving prediction accuracy. The A-DNNABF algorithm is designed to output a weight vector that

approximates the MVDR algorithm. Theoretically, we can represent this approximation as:

$$\left\| W_{A-DNNABF} - w_{MVDR} \right\| < \varepsilon$$

where $w\_\{A - DNNABF\}$ is the weight vector output by the A-DNNABF algorithm, w_{MVDR} is the ideal weight vector of the MVDR algorithm, and ε is a small positive number representing the approximation error.

Answer to RQ2: Our method, based on an optimized Transformer architecture, significantly outperforms the DNNABF algorithm in terms of prediction accuracy. This result not only verifies the effectiveness of our method but also provides a new high-precision solution for the field of adaptive beamforming. Particularly in complex and variable signal environments, our method shows stronger generalization ability and lower prediction error. This high-precision prediction has the potential to provide more reliable beamforming performance in practical applications, such as improving signal quality in communication systems or enhancing target detection capability in radar systems. However, we also acknowledge that while excellent performance was achieved on test data, broader experimental validation is needed before applying this method to real systems, especially evaluating its stability and robustness under various signal environments and interference conditions.

RQ3. Does our algorithm possess the strongest robustness?

The primary motivation for studying the robustness of algorithms stems from the complexity and variability of real-world signal-processing environments. The number of interference sources may change in practical applications, and an ideal algorithm should maintain stable performance under such variations. Robustness here specifically refers to the algorithm's ability to accurately estimate the angles of arrival for both the desired and interference signals, even when the number of interference sources changes. High robustness means the algorithm can adapt to different signal environments and provide reliable beamforming performance. Our research aims to assess whether the method based on an optimized Transformer architecture can outperform existing DNNABF algorithms in this regard, offering more reliable solutions for complex and variable signal processing scenarios.

To objectively evaluate the algorithm's robustness, we adopted two key metrics: the absolute average angular deviation of the desired signal $\left|\overline{\Delta\theta_0}\right|$ and the absolute average angular deviation of the interference signal $\left|\overline{\Delta\theta_{null}}\right|$. These metrics directly reflect the accuracy of the algorithm in estimating the angles of arrival for signals. The absolute average angular deviation of the desired signal is calculated as

$$\left|\overline{\Delta\theta_0}\right| = \sum_{i=1}^{S_0} \left| \theta_0^{(i)} - \hat{\theta}_0^{(i)} \right| / S_0 \quad \textbf{(10)}$$

where S_0 is the total number of desired signals, and $\theta_0^{(i)}$ and $\hat{\theta}_0^{(i)}$ are the actual and estimated values, respectively. Similarly, the absolute average angular deviation of the interference signals is calculated as

$$\left|\overline{\Delta\theta_{null}}\right| = \sum_{j=1}^{S_{null}} \left| \theta_{null}^{(j)} - \hat{\theta}_{null}^{(j)} \right| / S_{null} \quad \textbf{(11)}$$

The smaller these metrics, the stronger the robustness of the algorithm, indicating its ability to maintain high-accuracy angle estimations despite changes in the number of interference sources. The experimental results are shown in Table 1.

Answer to RQ3: Our proposed method, based on the optimized Transformer architecture, significantly outperforms the DNNABF algorithm in terms of robustness. This result not only validates the effectiveness of our approach but also offers a more reliable solution for complex and dynamic signal processing environments. Particularly in application scenarios where the number of interference sources may change frequently, such as in mobile communications or dynamic radar systems, our method demonstrates stronger adaptability and more stable performance. This high robustness has the potential to improve the reliability and efficiency of systems in practical applications, such as in wireless communication in complex urban environments or in military applications with severe electromagnetic interference.

Experimental results show that for a varying number of interference signal sources from 3 to 8, our proposed method based on the optimized Transformer architecture consistently exhibits significantly lower average error for both the expected signal and interference signal compared to the DNNABF algorithm. This result can be explained from several perspectives. Firstly, the self-attention mechanism of the Transformer architecture possibly makes it more adaptable to structural changes in input signals, effectively capturing inter-signal relationships regardless of the number of interference sources. Secondly, our method is optimized through architecture search, potentially identifying a network structure that maintains stable performance under varying interference conditions. Furthermore, the position encoding mechanism of the Transformer might better preserve the spatial information of signals, accurately locating signal sources even when the number of interference sources changes. Finally, our training strategy may have incorporated more diverse interference scenarios, enhancing the model's generalization capability.

Table 1. Comparison of error for different algorithms with varying number of interference sources

Number of interferences	$\overline{\Delta\theta_0}$		$\overline{\Delta\theta_{null}}$	
	DNNABF	A-DNNABF	DNNABF	A-DNNABF
3	0.04	0.021	1.08	0.973
4	0.09	0.075	1.255	1.148
5	0.09	0.064	1.412	0.955
6	-	0.063	-	1.579
7	-	0.078	-	1.325
8	-	0.067	-	1.457

6 Conclusion and Future Research Directions

6.1 Conclusion

The A-DNNABF algorithm proposed in this study has achieved significant results in adaptive beamforming under multiple noisy signal environments. By integrating automated neural architecture search with attention mechanisms, we successfully developed an algorithm capable of efficiently handling rapidly changing signal environments. The experimental results convincingly demonstrate the superiority of the A-DNNABF algorithm in several key areas: Compared to the traditional MVDR algorithm, A-DNNABF improves computational speed by about tenfold, enabling real-time signal processing; relative to the DNNABF algorithm, A-DNNABF exhibits lower prediction errors across datasets of different sizes, proving its excellent generalization ability; when the number of interference sources changes, A-DNNABF consistently maintains low angle estimation error for desired and interference signals, demonstrating strong environmental adaptability; through automated learning of network architecture, A-DNNABF effectively adapts to the rapid changes in the Angle of Arrival (AOA) of incoming signals, achieving real-time formation of beams and nulls. These results indicate that the A-DNNABF algorithm successfully addresses the response speed issue faced by traditional algorithms in rapidly changing environments while maintaining high accuracy. This research provides a novel solution for real-time adaptive beamforming in multiple noisy signal environments, holding significant application value in fields like modern communication systems and radar technology.

6.2 Future Research Directions

1) **Algorithm Optimization:** Further explore more advanced automated neural architecture search techniques and more efficient variants of attention mechanisms to enhance algorithm performance and computational efficiency.

2) **Robustness Enhancement:** Introduce adversarial training or more complex interference models to improve algorithm performance under extreme interference conditions.

3) **Multi-task Learning:** Explore multi-task learning frameworks that combine beamforming with other related tasks (such as channel estimation and signal classification) to improve the overall system performance.

4) **Hardware Implementation:** Investigate the implementation of the A-DNNABF algorithm on dedicated hardware (such as FPGA or ASIC) to further enhance real-time processing capabilities.

5) **Application Expansion:** Explore the potential applications of the algorithm in other fields, such as acoustic signal processing and medical imaging.

6) **Theoretical Analysis:** Conduct in-depth research into the theoretical foundations of the algorithm, including its convergence, stability, and generalization bounds, to provide theoretical guidance for further algorithm improvement.

7) **Dynamic Environment Adaptation:** Study how to enable the algorithm to learn and adapt online to cope with dynamically changing signal environments.

8) **Low-resource Scenarios:** Explore how to deploy and optimize the A-DNNABF algorithm in scenarios with limited computational resources, such as mobile devices or IoT devices.

Acknowledgment

This work is supported by the sixth “333 High-level Talents Training Project” in Jiangsu Province of China.

References

[1] S. Fiori, Neural minor component analysis approach to robust constrained beamforming, *IEE Proceedings-Vision, Image and Signal Processing*, Vol. 150, No. 4, pp. 205-218, August, 2003.

[2] A. B. Suksmono, A. Hirose, Intelligent beamforming by using a complex-valued neural network, *Journal of Intelligent & Fuzzy Systems*, Vol. 15, No. 3-4, pp. 139-147, 2004.

[3] W. Guo, T. Qiu, H. Tang, W. Zhang, Performance of RBF neural networks for array processing in impulsive noise environment, *Digital Signal Processing*, Vol. 18, No. 2, pp. 168-178, March, 2008.

[4] P. Ramezanpour, M. J. Rezaei, M. R. Mosavi, Deep-learning-based beamforming for rejecting interferences, *IET Signal Processing*, Vol. 14, No. 7, pp. 467-473, September, 2020.

[5] S. Singh, A. Sharma, State of the art convolutional neural networks, *International Journal of Performability Engineering*, Vol. 19, No. 5, pp. 342-347, May, 2023.

[6] V. Sudha, A. S. Vijendran, OSD-DNN: Oil spill detection using deep neural networks, *International Journal of Performability Engineering*, Vol. 20, No. 2, pp. 57-67, February, 2024.

[7] Y. Ren, Y. Du, J. Zhang, An adaptive beamforming algorithm based on deep neural network, *Telecommunication Engineering*, Vol. 62, No. 7, pp. 852-858, July, 2022.

[8] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.-L. Boulesteix, D. Deng, M. Lindauer, Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 13, No. 2, Article No. 1484, March/April, 2023.

[9] R. Bellman, R. Kalaba, On adaptive control processes, *IRE Transactions on Automatic Control*, Vol. 4, No. 2, pp. 1-9, November, 1959.

[10] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, *Journal of machine learning research*, Vol. 13, No. 2, pp. 281-305, February, 2012.

[11] D. R. Jones, M. Schonlau, W. J. Welch, Efficient global optimization of expensive black-box functions, *Journal of Global optimization*, Vol. 13, No. 4, pp. 455-492, December, 1998.

[12] J. Snoek, H. Larochelle, R. P. Adams, Practical bayesian optimization of machine learning algorithms, *Advances in neural information processing systems*, Lake Tahoe, Nevada, 2012, pp. 25-34.

[13] F. Hutter, H. H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, *5th international conference on Learning and intelligent optimization, LION 5*, Rome, Italy, 2011, pp. 507-523.

[14] H.-G. Beyer, H.-P. Schwefel, Evolution strategies–a comprehensive introduction, *Natural computing*, Vol. 1, No. 1, pp. 3-52, March, 2002.

[15] H.-G. Beyer, B. Sendhoff, Evolution strategies for robust optimization, *2006 IEEE international conference on evolutionary computation*, Vancouver, Canada, 2006, pp. 1346-1353.

[16] X. He, K. Zhao, X. Chu, AutoML: A survey of the state-of-the-art, *Knowledge-based systems*, Vol. 212, Article No. 106622, January, 2021.

[17] H. Liu, K. Simonyan, Y. Yang, Darts: Differentiable architecture search, *arXiv preprint*, arXiv:1806.09055, June, 2018. https://arxiv.org/abs/1806.09055

[18] J. Bergstra, D. Yamins, D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, *The 30th International conference on machine learning*, Atlanta, USA, 2013, pp. 115-123.

[19] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, Anchorage, AK, USA, 2019, pp. 2623-2631.

[20] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, I. Stoica, Tune: A research platform for distributed model selection and training, *arXiv preprint*, arXiv:1807.05118, July, 2018. https://arxiv.org/abs/1807.05118

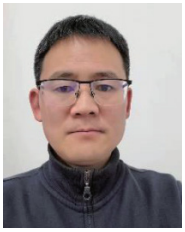
[21] T. Bartz-Beielstein, Hyperparameter tuning cookbook: a guide for scikit-learn, PyTorch, river, and spotPython, *arXiv preprint*, arXiv:2307.10262, July, 2023. https://arxiv.org/abs/2307.10262

[22] J. Capon, High-resolution frequency-wavenumber spectrum analysis, *Proceedings of the IEEE*, Vol. 57, No. 8, pp. 1408-1418, August, 1969.

Biographies

Zheng Xu received the M.S. degree from Soochow University in 2012. He is currently pursuing the Ph.D. degree in the Army Engineering University of PLA, China. He is also the chief designer in Nanjing Panda Handa Technology Co., Ltd., China. His research interests include beamforming technology and satellite communication.

Zihao Pan received the M.S. degree in electronic information from the Army Engineering University of PLA, Nanjing, China, in 2022, where he is currently pursuing the Ph.D. degree in communication and information system. His research interests include array signal processing, satellite communication, and channel estimation and equalization.



Daoxing Guo is currently a Full Professor with Army Engineering University of PLA, Nanjing, China. He received the M.S. degree and Ph.D. degree from Institute of Communications Engineering, Nanjing, China, in 1999 and 2002 respectively. He has authored and coauthored more than 40 refereed professional research papers.