IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 28 June 2025, accepted 15 July 2025, date of publication 22 July 2025, date of current version 29 July 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3591643

RESEARCH ARTICLE

Deep Reinforcement Learning-Based Adaptive Nulling in Phased Array Under Dynamic Environments

YING-DAR LIN[1], (Fellow, IEEE), JEN-HAO CHANG[1], AND YUAN-CHENG LAI[2], (Member, IEEE)
[1]Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan
[2]Department of Information Management, National Taiwan University of Science and Technology, Taipei 10607, Taiwan

Corresponding author: Yuan-Cheng Lai (laiyc@cs.ntust.edu.tw)

This work was supported in part by the National Science and Technology Council (NSTC), Taiwan, under Project MOST 111-2221-E-011-089-MY3; and in part by WavePro Inc.

ABSTRACT A phased array consists of multiple antenna elements that can control the direction of the radiated signal by adjusting each antenna element's phase and amplitude, which are encapsulated in the phased array weights. To obtain better communication quality, nulling, which can weaken the interference signal, is helpful by adjusting the phased array weights. In dynamic environments, rapid changes in the directions of both interference and desired signals demand equally rapid, continual updates of phased-array weights. Traditional heuristic optimizers—such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA)—struggle to keep up because their iterative searches depend on pre-computed configurations that are unrealistic to obtain on the fly. To date, no heuristic, supervised-learning, or reinforcement-learning method simultaneously achieves all three requirements: fast adaptation, dataset-free operation, and continuous complex-weight control under highly dynamic environments. In this paper, an innovative deep reinforcement learning-based adaptive nulling, called DRLNuller, is proposed. DRLnuller adopts the Proximal Policy Optimization (PPO) algorithm, a typical reinforcement learning algorithm, to dynamically optimize phased array weights through continuous interaction with the environment without relying on pre-computed or labeled data. In experiments, DRLNuller after the training process outperforms PSO and GA in computation speed by 2.83×10^5 times faster and maintains effective communication quality, an average Signal-to-Interference Ratio (SIR) of 25.06 dB, under different conditions.

INDEX TERMS Phased array, interference mitigation, deep reinforcement learning, adaptive nulling.

I. INTRODUCTION

A phased array is a system composed of multiple antenna elements, each capable of having its phase and amplitude individually adjusted. By precisely altering these parameters for each antenna element, it's possible to construct a specific radiation pattern, which is a graphical representation of the strength of signals emitted from or received by the phased array in different directions. This feature allows a phased array to redirect its focused emitted energy in different directions rapidly, providing a means to swiftly adjust the radiation pattern. This capability enables the system to communicate in a targeted direction, effectively focusing the majority of the energy towards the desired point. Such a design significantly enhances its overall efficiency, as it can be electronically steered to different directions without necessitating any physical movement of the components.

The radiation pattern produced by a phased array is influenced by a multitude of factors. Among these factors, array geometry, which includes the arrangement of antenna elements and their relative spacing, as well as the distance between each antenna element, is significant. However, the most critical factor is the phased array weights, referring to the specific phase and amplitude values applied to each antenna element. Adjust the phased array weights shapes its overall radiation pattern, which is synthesized from the

The associate editor coordinating the review of this manuscript and approving it for publication was Shah Nawaz Burokur.

contributions of each individual antenna element. In crafting this pattern, it is essential to consider the mutual effects of the antenna elements on each other. The precise determination of phase and amplitude settings enables a phased array to efficiently focus energy in the desired direction while minimizing energy dispersion towards directions of interference.

Phased array systems face performance degradation due to interference, which can stem from unintended or malicious sources. Normal interference includes extraneous signals like those from broadcast towers, inadvertently raising noise levels and diminishing communication quality by masking the intended signal. Conversely, malicious interference, such as jammers or fake signals, is designed to disrupt system operations, leading to erroneous system responses and compromised decision-making. Both types of interference threaten phased array system's effectiveness and reliability, highlighting the critical need for effective countermeasures to safeguard against such disruptions and ensure the system's optimal performance. Nulling is the method that suppress interference signals in phased arrays. Its critical function can be achieved by carefully controlling the weight of each antenna element within the array. The primary goal of this control is to diminish the signal strength in the direction of interference, thereby enhancing the overall communication quality.

Traditionally, the optimization of phased array weights relies on iterative algorithms that adjust the phase and amplitude of each antenna element through multiple iterations until they converge on suitable values [1], [2], [3], [4]. These iterative algorithms work effectively in static or slowly changing environments, where the direction of interference remains relatively stable and the algorithm has sufficient time to reach an optimal solution. However, in dynamic scenarios, such as airborne devices like military drones can travel at speeds up to 180 km/h, causing the interference direction changes continuously and rapidly. This speed can cause the angle shifts by 0.1° every millisecond at a distance of 30 meters because of $\frac{0.1\times\pi}{180^\circ} \times 30$ meters $=$ 0.05235 meters/ms $=$ 52.35 meters/s $=$ 188.46 km/h. As a result, these algorithms, due to their iterative nature, struggle to update the phased array weights quickly enough to maintain an optimal radiation pattern.

Recent research [5], [6], [7], [8], [9], [10] has delved into supervised Machine Learning (ML) techniques for phased array weight optimization, training neural network models with labeled datasets to predict optimal weights swiftly, thereby addressing the need for low time-complexity demands [9], [11], [12]. Despite the efficiency in inference, the approach stumbles upon the challenge of dataset creation, which hinges on pre-knowing optimal weights across various interference scenarios. This requirement becomes a stumbling block in real environments, where interference patterns are neither constant nor predictable, making comprehensive dataset compilation an arduous task.

Reinforcement Learning (RL) stands out as a robust method for dynamically optimizing phased array weights in fluctuating interference scenarios, eliminating the dependency on pre-existing datasets [13]. It learns and refines strategies through direct environment interaction, constantly enhancing its decision-making process. Thus, RL can offer a real-time solution for phased array optimization, excelling in complex, ever-changing environments with its efficient computational prowess.

In this paper, we introduced a method grounded in Deep Reinforcement Learning (DRL), called DRLNuller, which stands out by utilizing feedback directly from the environment. This direct interaction with the environment allows DRLNuller to dynamically adjust and optimize the weights of phased array in real-time, without the necessity for pre-existing training datasets and ensuring low inference time. The key to DRLNuller's effectiveness is its ability to learn and adapt autonomously, continuously enhancing the performance of a phased array by refining its strategy based on environmental feedback. The objective is to improve the Signal-to-Interference (SIR) by adjusting the phased array weights, so DRLNuller treats this value as its reward function.

DRLNuller adopts a typical DRL algorithm, Proximal Policy Optimization (PPO) [14], which stands out due to its simplicity, stability, and versatility. PPO employs a clipping mechanism that balances exploration and policy stability, making it easier to implement and computationally efficient. It is highly sample-efficient, reusing on-policy data effectively, and works well in both discrete and continuous action spaces. PPO also offers robust performance across various tasks, requires less fine-tuning of hyperparameters, and delivers a reliable trade-off between performance and training stability, making it a preferred choice for many applications.

Contrary to previous RL-based nulling methods, our proposed DRLNuller has some unique aspects: (1) it generates complex weights directly from the model while previous work generates amplitude/phase within discrete libraries; (2) It learns online from raw signal feedback without any labelled dataset while previous work seeds the model with a small set of labeled examples so it gets a rough sense of what to do when it encounters unlabeled data; and (3) it delivers fast inference through a RL architecture with a six-layer actor network while the previous studies only consider accuracy, but doesn't take inference speed into account. These three properties make DRLNuller the first RL-based nuller suitable for deployment on airborne or vehicular platforms, where interference angles can change rapidly.

The principal contributions are as follows. (1) We formulate phased-array nulling under continuously and unpredictably varying interference, abandoning the common assumption that environmental changes are either slow or known a priori. (2) We develop DRLNuller, which applies PPO to learn end-to-end complex-weight control, thereby eliminating look-up tables and other pre-computed artefacts. (3) We carry out a comprehensive evaluation that contrasts DRLNuller with a reinforcement-learning baseline and heuristic algorithms, reporting both inference latency and

signal-to-interference ratio (SIR) alongside ablation studies on key hyper-parameters. Together, these results demonstrate that DRLNuller meets the sub-millisecond nulling demands of dynamic phased-array systems while delivering an order-of-magnitude reduction in computation time relative to traditional methods.

This work is organized as follows. Section II presents the necessary background and related work. Section III introduces our problem statement, laying the foundation for discussing our approach in Section IV. Section V details how to simulate the proposed approach. Section VI evaluates the solution through many experiments. Finally, Section VII concludes the paper by summarizing our findings and suggesting some future directions.

II. BACKGROUND AND RELATED WORKS

This section first presents the necessary background of phased array, common heuristic algorithms used in nulling, and reinforcement learning. The related work on nulling are also surveyed to differentiate our research with the existing literature.

A. PHASED ARRAY

A phased array is a system comprised of multiple antenna elements, each of which can have its phase and amplitude meticulously adjusted. This fine-tuning allows a phased array to construct a specific radiation pattern, enabling the control of signal power in a designated direction. When the power directed towards a specific direction is intensified, the signal in that direction is enhanced, leading to improved communication quality or detection capability. Conversely, if the power is diminished, the signal in the targeted direction is suppressed, which can be beneficial for minimizing interference or avoiding detection in radar applications.

The manipulation of antenna elements is facilitated by antenna weights, which are represented by complex numbers. These complex weights encapsulate both the phase and amplitude adjustments necessary for the desired signal manipulation. The phase and amplitude of each complex weight, $\omega = R + jI$, are obtained as

$$\begin{aligned} \text{phase} &= \text{atan2}(R, I) \times \frac{180}{\pi} \text{ and} \\ \text{amplitude} &= 10 \log_{10}(R^2 + I^2), \end{aligned} \tag{1}$$

where R and I are the real and imaginary parts of the weight. It's possible to convert these complex weights into the phase and amplitude adjustments required for the phased array, thus achieving the intended radiation pattern and signal control.

B. HEURISTIC ALGORITHMS

Phased array weights can be optimized using heuristic algorithms [15], [16], [17], which should be compared with our proposed approach. In the realm of heuristic algorithms, Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) offer distinct advantages. PSO is beneficial due to its relatively simple operations and fewer parameters requiring adjustment, while GA excels at exploring the global solution space and avoiding convergence to local optima through its genetic diversity mechanism.

In PSO, each solution to the optimization problem is represented as a particle in the swarm [16]. The particles move through the search space, guided by their personal best positions and the best positions found by the swarm, to find the optimal solution. It starts with a swarm of particles, initialized with random positions and velocities. Each particle evaluates its position using a fitness function, updating its personal best position if it finds a better solution. The global best position is also tracked across the swarms. A particle adjusts its velocity based on three components: its current velocity, a cognitive component influenced by its personal best position, weighted by a factor c_1, and a social component influenced by the global best position, weighted by a factor c_2. Random factors are also introduced to ensure diversity. Particles then update their positions. This process continues until a stopping criterion is satisfied, such as reaching a maximum number of iterations or achieving an acceptable fitness level.

GA mimics biological evolution processes such as selection, crossover, and mutation to evolve a population of solutions toward an optimal solution [17]. It begins with a randomly initialized population of individuals, each encoding a potential solution. The fitness of each individual is evaluated, and individuals are selected for reproduction based on their fitness. Selected individuals undergo crossover, exchanging chromosome segments to produce offspring, followed by mutation, where random changes are introduced to maintain diversity. The new generation replaces the old one, continuing this cycle until a stopping criterion is satisfied, such as reaching a maximum number of generations or achieving a satisfactory fitness level. The final population contains the optimized solutions.

C. REINFORCEMENT LEARNING

In RL, its operational framework comprises the Environment, Agent, State, Action, Reward, and Policy. The agent explores the environment, aiming to achieve a specific goal. This goal is typically quantified as the attainment of the maximum cumulative reward, serving as a beacon guiding the agent's actions. To navigate this landscape, the agent relies on the state information provided by the environment and decides on actions that could be either discrete or continuous. The agent selects an action with the intention of deriving the maximal reward. The reward mechanism serves as immediate feedback from the environment, evaluating the efficacy of the agent's actions. The crux of RL lies in discovering and optimizing the policy that maximizes rewards for every possible state.

Given the diversity in problems addressed by RL, selecting an RL algorithm is crucial and should be tailored to the problem at hand, ensuring the most effective solution is employed. In the field of phased array weight optimization, efficiently navigating the complex and continuous action spaces presents

a significant challenge. It is better to employ the Proximal Policy Optimization (PPO) [14] algorithm, situated within the actor-critic framework, which is particularly suited for such intricate tasks. Notable for its clipped objective function, PPO curtails drastic policy shifts, thereby fostering stable learning. This method is distinguished by its reduced need for environmental interaction, speeding up the adaptation process in dynamic settings for the phased array that requires fast responses to a changing environment. The actor-critic model used in PPO benefits phased array optimization by providing precise feedback for policy adjustments, facilitating faster convergence to optimal configurations. Furthermore, its ability to balance exploration and exploitation is crucial for avoiding local optima in the high-dimensional search spaces typical of phased array weight optimization.

The diagram in Figure 1 illustrates the training workflow of the PPO algorithm. Each interaction between the actor and the environment generates a transition that includes the current state s, action a, reward r and subsequent state s'. This transition is stored in a structure known as the rollout buffer. When the rollout buffer reaches capacity, the process of calculating errors for updating the neural networks of both the actor and the critic begins. For the critic, the value loss is computed using the Mean Squared Error (MSE) between states s and s', applying Temporal Difference (TD) learning with a discount factor to control the impact of s' on the value error [18]. For the actor, the training involves two types of losses, clip loss and entropy loss. Initially, the advantage value is calculated using Generalized Advantage Estimation (GAE), which incorporates a GAE lambda parameter to balance bias and variance. Subsequently, the probability ratios are used to compute the clip loss. After obtaining both the advantage value and the ratios, the clip loss is determined within a specified range, and the entropy loss is calculated based on the current policy. The total loss, which is a combination of the value loss, clip loss, and entropy loss, is then used to update the networks of both the actor and the critic. The contributions of value loss and entropy loss to the total loss are adjusted by their respective coefficients.

D. RELATED WORK

As shown in Table 1, the work related to this paper is under different environments, phased array and multiple input multiple output (MIMO). A phased array relies on the principle of superimposing waves from multiple antenna elements, adjusting the phase and amplitude of each to steer the energy direction without physically moving the antennas. Conversely, MIMO exploits spatial diversity by transmitting different data streams simultaneously across multiple antennas, thereby increasing the capacity and reliability of wireless communication channels.

The nulling technique can be categorized into two distinct types, deterministic and adaptive [19]. The deterministic approach utilizes predefined phased array weight settings, which are selected based on the specific characteristics of the interference situation. This method relies on a prior knowledge of the interference landscape, allowing for a quick but less flexible response. On the other hand, the adaptive approach leverages real-time feedback to dynamically adjust the phased array weights, aiming to optimally null the interference signal. This method offers greater flexibility and can respond to changing interference conditions, ensuring that a phased array maintains optimal performance by continuously adapting its configuration to a dynamic environment.

The approaches vary based on different inputs. The term ''phased array configuration'' refers to the setting of a phased array, such as its geometrical arrangement and the electronic characteristics or prior weights of the antenna elements. These factors determine the array's capabilities and directionality. ''Desired pattern characteristics'' pertains to the intended radiation pattern of the array, which can include numerical data like the main direction, main energy power or a graph of the radiation pattern. ''Autocorrelation matrix'' is a mathematical construct that captures the temporal and spatial attributes of incoming signals. It also offers insights into how the signal received by one antenna element relates to signals received by others.

In various studies, the output can be categorized into ''phase'', ''amplitude'', and ''complex weight''. ''Phase'' relates to the timing applied through the weight to each antenna element, essential for accurately directing the focus energy. On the other hand, output ''amplitude'' refers to the power level emitted by each antenna element, crucial for controlling the energy intensity produced by the phased array. ''Complex weight'', which is the complex number combines both phase and amplitude adjustments, offering a comprehensive control mechanism over the phased array.

The performance metric used in previous literature includes ''Signal-to-Interference'' (SIR) and ''null depth''. SIR is the ratio of the power of the desired signal to the power of interference signals. It measures how well a system can distinguish the desired signal from unwanted interference. On the other hand, null depth refers to how deep a antenna array can suppress signals in a specific direction, typically the direction of interference. SIR can be considered a global performance metric, whereas null depth is a directional suppression measure.

Recent research in the field of optimizing phased array can be broadly categorized into three main streams, Heuristic Algorithm, Supervised Learning and Reinforcement Learning. Heuristic Algorithm methods have been extensively applied to optimize the phased array, demonstrating significant advancements in this domain. Particle Swarm Optimization (PSO), an iteration-based heuristic approach, has been used due to its efficiency in navigating the complex solution space of phased array systems [1]. However, the iterative nature of PSO inherently requires longer calculation times to converge to an optimal solution. This characteristic becomes particularly problematic in dynamic environments where the direction of the signal is persistently changing. Under such circumstances, the PSO algorithm struggles to

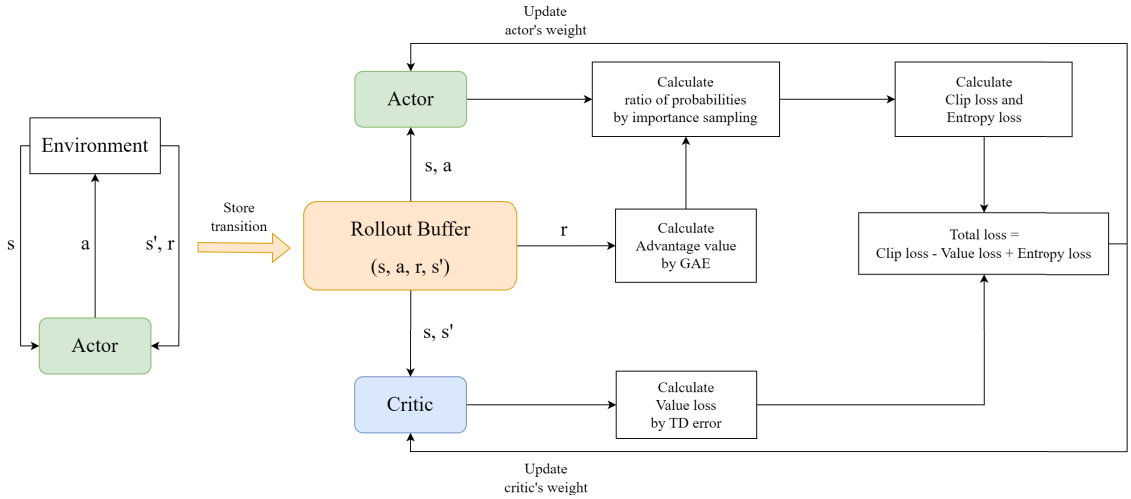

FIGURE 1. PPO training flow.

TABLE 1. Summary of optimizing antenna weights for nulling research.

Approach Type	Study	Environment	Nulling Type	Input	Output	Dataset	Method	Performance Metric
Heuristic Algorithm	[1]	PA	D	Phased Array configuration	Phase	X	PSO	*Avg. SIR* ≈ 67 *dB*
Supervised Learning	[11]	PA	A	Desired pattern characteristics	Complex weight	O	GRNN	*Null depth* ≈ -45 *dB*
	[12]	PA	A	Autocorrelation matrix	Complex weight	O	CNN	*Null depth* < -60 *dB*
	[20]	PA	A	Direction of desired / interference signal	Complex weight	O	LSTM	*Avg. SIR* ≈ 17.9 *dB*
	[21]	PA	A	Direction of desired / interference signal	Complex weight	O	GRU	*Avg. SIR* ≈ 17.8 *dB*
Reinforcement Learning	[22]	MIMO	D	Phased Array configuration	Phase	X	DDPG	*Avg. SIR* ≈ 18 *dB*
	[13]	PA	D	Desired pattern characteristics	Amplitude / Phase	O	Q-Learning	*Null depth* ≈ -40 *dB*
	Ours	PA	A	Direction of desired signal / Autocorrelation matrix	Complex weight	X	PPO	*Avg. SIR* ≈ 25.06 *dB*

PA: Phased Array, MIMO: Multi Input Multi Output; D: Deterministic, A: Adaptive

adapt quickly enough, making it difficult to produce a timely solution that can effectively respond to the rapid changes in signal direction.

A notable shift to supervised machine learning for fast inference has been observed [11], [12], [20], [21]. The study in [11] creatively employed optimization algorithms to generate labels for the training dataset, showcasing an innovative approach to supervised learning in this context. The study in [12] used a similar approach, but adopted an autocorrelation matrix, rather than desired pattern characteristics used in [11]. On the other hand, studies in [20] and [21] introduced models that take into account the relationship between current and previously received signals, offering a more nuanced understanding of signal dynamics. However, a fundamental challenge in employing supervised machine lies in the necessity to construct sufficiently large and well-labeled datasets.

To tackle the issue of feasibility, RL strategies has been adopted [13], [22]. Specifically, the study in [22] applied RL within the MIMO systems. In this approach, both the predicted signal and the array configuration are utilized as the state, enabling the system to select an appropriate phase setting from a predetermined set of weights and act as the action space in the RL framework. Similarly, the study in [13] focused its application on the phased array. This method allows to select suitable phase and amplitude settings from predetermined options for tailoring the phased array's output to specific requirements. Crucially, both studies leverage pre-calculated phased array weights as the action set, emphasizing the need for a pre-processing step to transform the input state into a format conducive for the RL algorithm.

Our research presents an innovative approach by utilizing the desired direction and raw signal data, sourced directly from a simulated environment, to directly confront the challenges associated with dataset construction and the generalization problem. By leveraging a neural network as the reinforcement learning (RL) agent, our method diverges from traditional strategies that rely on predetermined settings. Instead, our neural network agent dynamically generates its output actions based on real-time feedback from the environment. This direct interaction with the simulated environment allows the neural network to adaptively learn

and adjust its strategies over time, offering a more flexible and responsive solution compared to statically predefined action sets. Thus this methodology not only mitigates the limitations posed by the need for an extensive labeled dataset but also enhances the model's ability to generalize across a broader spectrum of scenarios, showcasing a significant advancement in the application of RL to optimize the phased array.

The main difference compared to [13] and [22] mainly lies in two points. The first is the way actions selected. Our method directly searches within a continuous solution space, whereas [13] and [22] choose an action from a pre-quantized and discrete weight set. Using this weight set limits the nulling capability of a phased array, as the antenna weights are pre-quantized and therefore cannot provide the optimal weights for a given input. The second is the input data used. Our input uses raw signals, while [13] relies on relatively high-level features (radiation patterns), which may lead to degraded performance when higher directional precision is required. On the other hand, [22] requires estimating the interference angle at every step, and any estimation error propagates to the next layer of the RL model, thereby degrading overall performance. In contrast, our approach uses low-level features, raw signals, and outputs continuous complex weights, with training a single end-to-end model that achieves high-resolution nulling performance without being affected by angle estimation errors.

III. SYSTEM MODEL AND PROBLEM STATEMENT

This section first introduces the system model and defines notations used. Afterwards, the formal problem is stated.

A. SYSTEM MODEL

The system model is shown in Figure 2. The used notations, including the categories of phased array, signal, and RL, are shown in Table 2. The notations about phased array and signal are introduced in this section and those about RL will be explained in the next section.

The receiver, such as a drone, is designed to maintain a robust communication link with a ground-based station equipped with a phased array system. The phased array is crucial for its ability to dynamically focus the energy toward the desired signal while nulling out interference, thereby enhancing the communication quality in a dynamic environment. In our scenario, we consider a single interference source, such as a helicopter with a jamming device, which serves to simplify the initial model and focus on evaluating the effectiveness of DRL strategies in a controlled interference environment. This setup ensures that any observed improvements or issues can be attributed directly to the DRL strategies, rather than confounding factors introduced by multiple interference sources.

At time t, the phased array PA receives three types of signals: (1) the desired signal S_t^D from the receiver in direction θ_t^D. This is the signal that the phased array intends to receive. (2) The interference signal S_t^I from an interference source that the phased array aims to exclude in direction θ_t^I. (3) Gaussian noise S_t^N, which is uniformly distributed in all directions. Thus, the final received signal S_t^R at the phased array is the summation of S_t^D, S_t^I and S_t^N.

Based on S_t^R, optimizing phased array weight module will calculate the phased array weight PA_t^W, which includes the complex weights $A_{i,t}^W$ for each antenna element $A_i (1 \leq i \leq P^{NA})$, where A_i denotes the i-th antenna and N^{NA} is the number of antennas. Applying PA_t^W to a phased array system constructs a radiation pattern optimized to enhance the desired signal power P_t^D in the direction θ_t^D while reducing the interference signal power P_t^I in the direction θ_t^I.

B. PROBLEM STATEMENT

The problem is to optimize phased array weights, PA_t^W. The input comprises the desired signal direction θ_t^D and the received signal S_t^R. The output from this optimization process is a set of antenna element weight $A_{i,t}^W$. The main objective is to maximize the Signal-to-Interference Ratio SIR_t, defined as the difference between the desired signal power P_t^D and the interference signal power P_t^I. We use the subtraction, rather than the division, because the units of both P_t^D and P_t^I have already been converted to dB. This optimization problem is subject to constraints, $P_t^D > 0$, ensuring that the SIR is sufficient to maintain reliable communication without degradation of the signal integrity, and the $P_t^I < \eta^{RL}$, where η^{RL} is the parameter that constrains the interference signal. This effectively mitigates interference signal, thereby reducing the influence of interference on the performance of the phased array.

- **Input**
  - Desired signal direction, θ_t^D.
  - Received signal, S_t^R.
- **Output**
  - Complex weights of phased array PA_t^W, where $PA_t^W = [A_{1,t}^W, \cdots, A_{i,t}^W, \cdots, A_{N^{PA},t}^W]$.
- **Objective**
  - Maximize Signal-to-Interference Ratio (SIR), defined as $SIR_t = P_t^D - P_t^I$ (dB).
- **Constraints**
  - Upper bound of interference signal power: $P_t^I < \eta^{RL}$.
  - Lower bound of desired signal power: $P_t^D > 0$.

IV. DRLNuller

In this section, we present DRLNuller, which is designed to optimize the weight of phased arrays for adaptive nulling in dynamic environments using DRL. The core idea behind DRLNuller is to employ the PPO algorithm within a actor-critic framework. This allows the system to continuously interact with the environment and learn optimal strategies for adjusting phased array weights in real time.

In the proposed solution, the DRLNuller model M^{RL} is meticulously trained to optimize the phased array weight PA_t^W. As shown in Figure 3, the training process begins with the model receiving the signal S_t^R from the environment.

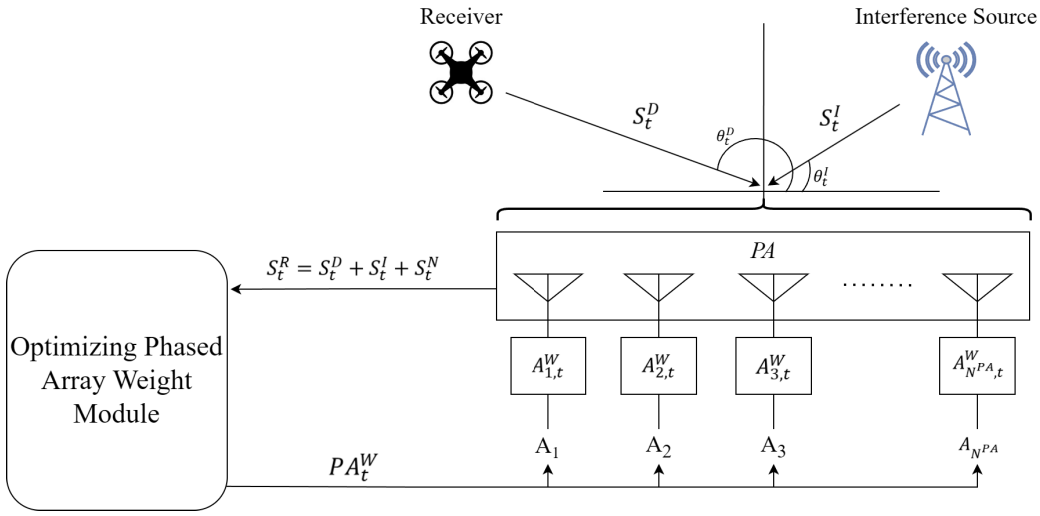

FIGURE 2. System model.

TABLE 2. Notations.

Symbol	Description
	Phased Array
PA	Phased array
N^{PA}	Number of antennas in phased array
A_i	The i-th antenna, where $i \in [1, \cdots, N^{PA}]$
$A_{i,t}^W$	The complex weight of the i-th antenna at time step t
PA_t^W	The complex weights of phased array at time step t, where $PA_t^W = [A_{1,t}^W, \cdots, A_{N_{PA},t}^W]$
	Signal
θ_t^D	The direction of desired signal at time step t, where $\theta_t^D \in [0°, 180°]$
θ_t^I	The direction of interference signal at time step t, where $\theta_t^I \in [0°, 180°]$
S_t^D	The signal attribute vector come from θ_t^D at time step t
S_t^I	The signal attribute vector come from θ_t^I at time step t
S_t^N	The random noise in the environment at time step t, sampling from Gaussian distribution
S_t^R	The final signal send into phased array at time step t, where $S_t^R = S_t^D + S_t^I + S_t^N$
P_t^D	The ratio of energy strength between the input and output of the desired signal at time step t. The unit is dB
P_t^I	The ratio of energy strength between the input and output of the interference signal at time step t. The unit is dB
SIR_t	The ratio of desired direction signal power to interference direction signal power at time step t. The unit is dB
	Reinforcement Learning
M^{RL}	The neural network trained by RL
Γ_t^{RL}	The autocorrelation matrix, post-processing of S_t^R at time step t
S_t^{RL}	The input of the RL model at time step t, where $S_t^{RL} = [\theta_t^D, \Gamma_t^{RL}]$
A_t^{RL}	The output of the RL model at time step t, where $A_t^{RL} = PA_t^W$
F^{RL}	The reward function used by the RL model
η^{RL}	The constraint on P_t^I. It is used to control the threshold for issuing a penalty reward
R_t^{RL}	The reward used to update the RL model at time step t, where $R_t^{RL} = F^{RL}(P_t^D, P_t^I)$
I^{RL}	The average calculation time of the RL model (the inference time)
K^{RL}	The average convergence time of the RL model (the training time)

This signal is then processed to calculate the autocorrelation matrix Γ_t^{RL}, which captures the spatial characteristics of the received signal. This matrix enhances the neural network's understanding of the interrelationships between the received signal and each antenna element. The autocorrelation matrix, once concatenated with the desired signal direction θ_t^D, forms the comprehensive input S_t^{RL} for M^{RL}. This input storing information about the received signal allows the neural network to reveal more subtle correlations and patterns in the data, which is beneficial for conducting in-depth data analysis and providing decision support.

We compute the autocorrelation matrix from the synchronized complex baseband signals across all antenna elements. However, in practice, hardware phase errors and calibration drifts are inevitable. Fortunately, minor phase mismatches are mitigated for three reasons. First, the phase errors of the interference signal are usually random, their impact on the autocorrelation matrix may be averaged out,

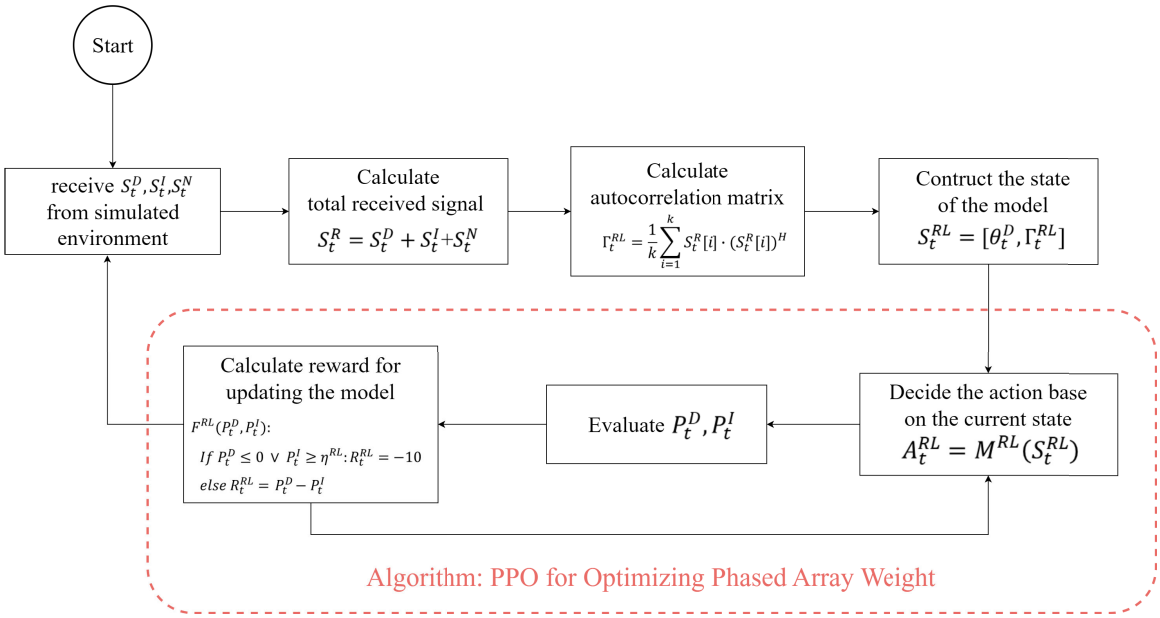

FIGURE 3. DRLNuller training process.

resulting in a relatively small effect. Second, we apply normalization or scaling to the measured autocorrelation matrix, which reduces the influence of small phase and amplitude deviations. Finally, our RL framework continually adjusts the phased array weights based on real-time signal-to-interference ratio feedback. Thus, even with moderate phase errors, the DRL agent effectively learns to compensate and maintain performance.

Follow Algorithm 1. In PPO, two neural networks are employed, the actor network π_ϕ and the critic network Q_θ. Additionally, a rollout buffer B is used to store interactive data, which aids in increasing sample efficiency and stabilizing actor network.

The state (s) is defined by a combination of autocorrelation matrix Γ_t^{RL} and the desired direction θ_t^D. The actor model determines the phased array weights (a) based on current state (s). Once these weights are applied to the phased array, the environment generates the next state (s') and a reward r. This historical data is then stored in the rollout buffer B as a transition tuple (s, a, r, s') and will later used to calculate the advantage estimation and update both actor and critic networks.

Once the size of the rollout buffer reaches its maximum capacity C, we estimate the advantage value used for the actor loss using the General Advantage Estimation (GAE) method. The actor loss comprises two parts, Clip loss and Entropy loss. Clip loss prevents the policy updates from being too drastic compared to the old policy, while Entropy loss encourages the policy to explore better actions. The critic loss is calculated as the Mean Square Error (MSE) between the estimated current state value and the sum of the reward and the estimated next state value, enhancing the critic's accuracy in estimating state values. After calculating the actor and critic losses, we use the Adam algorithm to update the weights ϕ of the actor network and the weights ω of the critic network. Finally, we clear the rollout buffer for new transitions.

Algorithm 1 PPO for Optimizing Phased Array Weight

1: Initialize actor network π_ϕ, critic network Q_ω
2: Initialize rollout buffer B with maximum size C
3: **for** $t = 1, 2, \ldots$ **do**
4: State $s = S_t^{RL}$ is equal to the concatenation of Γ_t^{RL} with θ_t^D
5: Select action $a = A_t^{RL} = \pi_\phi(s)$ by inputting s into current policy π_ϕ
6: Using MATLAB function "ArrayGain()" to evaluate P_t^D and P_t^I
7: Calculate reward $r = R_t^{RL} = F^{RL}(P_t^D, P_t^I)$ in lines 8 $\sim$ 12
8: **if** $P_t^D <= 0 \vee P_t^I >= \eta^{RL}$ **then**
9: $r = -10$
10: **else**
11: $r = P_t^D - P_t^I$
12: **end if**
13: Construct next state $s' = S_{t+1}^{RL}$
14: Store transition tuple (s, a, r, s') in B
15: $s \leftarrow s'$
16: **if** $size(B) = C$ **then**
17: Calculate the advantage estimation by GAE
18: Calculate the clip loss, value loss and entropy loss
19: Update ϕ and ω by Adam
20: Clear B
21: **end if**
22: **end for**

The reward function F^{RL} is defined from lines 8 to 12 in Algorithm 1. This function evaluates the difference between the power in the desired and interference directions. However, if $P_t^D \leq 0$, indicating that the desired signal was not enhanced or even suppressed, or if $P_t^I \geq \eta^{RL}$, indicating that the interference signal was not suppressed below the threshold or was even enhanced, the reward is then set to -10 directly. Because a reward of -10 is a serious penalty compared to the positive rewards which range between 0 and 1, it is sufficient to let the model know what a bad action is and help the model avoid this punishment. From the experiment later, values milder than -5 do not sufficiently discourage unsafe actions, while harsher values less than -20 dominate the return signal and slow convergence.

We assume that the directions of both the desired and interference signals are estimated using conventional algorithms prior to the nulling procedure. Although small errors in these estimates may arise due to noise or environmental factors, our proposed DRLNuller continuously adapts through interaction with the environment. Minor inaccuracies gradually incur lower rewards, prompting the RL agent to adjust the antenna weights and thereby compensate for these mismatches, preserving robust nulling performance.

Through iterative adjustments driven by reinforcement learning's reward mechanism, M^{RL} refine its policy to adeptly navigate the dynamic environment, leading the phased array towards better performance.

V. SIMULATION

In this section, we first introduce the open sources and tools used in the simulation. Afterwards, we briefly describe the tested architecture.

A. OPEN SOURCES AND TOOLS

Table 3 lists the primary open-source tools and libraries utilized in this study. The simulation of the received signal and the operation of a phased array are conducted using MATLAB and its Phased Array Toolbox plugin. The deep reinforcement learning algorithm is implemented with PyTorch, a Python deep learning framework, supplemented by Optuna for hyperparameter tuning. To enable interaction between Python code and MATLAB, the matlabengine Python library is employed, which facilitates data type conversion between Python and MATLAB. Traditional optimization algorithms are implemented using pymoo, which supports the integration of various heuristic algorithms through the definition of parameters like variable bounds, objective functions, and constraints. For data visualization, the libraries Seaborn and Matplotlib are employed.

B. TESTBED

The testbed architecture is shown in Figure 4. It comprises three main components, the phased array simulated environment, the agent, and the transformation part. The agent is responsible for optimizing the phased array weights using either heuristic algorithms or PPO. The phased array simulated environment simulates the received signal and the operation of the phased array, such as applying weights or calculating the power in a specific direction. The agent calculates the current weight based on environmental conditions using either heuristic algorithms or a deep reinforcement learning model. The transformation component handles data passing and data type conversion between the environment and the agent.

The testbed operates on the Ubuntu 22.04.1 LTS system. The hardware setup includes an Intel Core i7-8700 CPU clocked at 3.20 GHz and an NVIDIA GeForce RTX 3060 GPU with 12 GB of VRAM. This setup is supported by Python 3.10.13, MATLAB R2023b, and CUDA 12.2.

In the phased array simulated environment, we utilize the MATLAB programming language. To simulate the signal, we first generate a pulse wave and then use the "collectPlaneWave()" function from the MATLAB Phased Array Toolbox to create signals in specific directions, including both desired and interference signals. Additionally, we sample noise from a Gaussian distribution. By combining the desired signal, interference signal, and noise, we obtain the final received signal that is then passed to the agent.

The evaluation of the agent's actions within the phased array simulated environment is implemented using the "ArrayGain()" function from the Phased Array Toolbox to compute the power of signals in specific directions. The results are then returned to the agent to calculate the reward, which is used to update the deep reinforcement learning model.

In the transformation component, we used matlabengine, a Python library, to convert data types from Python to MATLAB and pass parameters to specific functions. This process enabled us to utilize MATLAB's advanced computational functions directly from our Python scripts.

In the PPO implementation, we use Python and PyTorch to develop the deep reinforcement learning model and the PPO algorithm. Since the phased array weights are complex numbers, the output of the neural network model will be twice the number of antenna elements, containing the real and imaginary parts of the weights.

VI. EVALUATION

In the evaluation, we discuss four key issues. First, we compare the performance of DRLNuller, another RL algorithm called Deep Deterministic Policy Gradient (DDPG) [22], and heuristic algorithms, i.e., PSO and GA. This comparison evaluates how DRLNuller effectively optimizes phased array weights and identifies its advantages and disadvantages in time-limited environments. Second, we examine the impact of phased array size, i.e., the number of antennas, P^{NA}, on the performance of DRLNuller. With a larger number of antennas in the phased array, the complexity of optimizing antenna weights increases, making the exploration of the solution space more challenging for DRLNuller. Third, we analyze how different signal direction variation scenarios

TABLE 3. Open source and tools.

Category	Name	Functionality
Simulation	MATLAB(R2023b)	A programming language that often used to simulate system or environment
	Phased Array Toolbox	A plugin in MATLAB. Provide callable function for simulating phased array operation
Library	Pytorch [23]	Mature deep learning framework in Python
	Optuna [24]	A python library for optimizing model's hyperparameter
	matlabengine	A Python library for connecting Python and MATLAB programs
	pymoo [25]	A Python library for quick implementation of optimization algorithms
	Seaborn [26], Matplotlib [27]	A Python library for visualizing the data

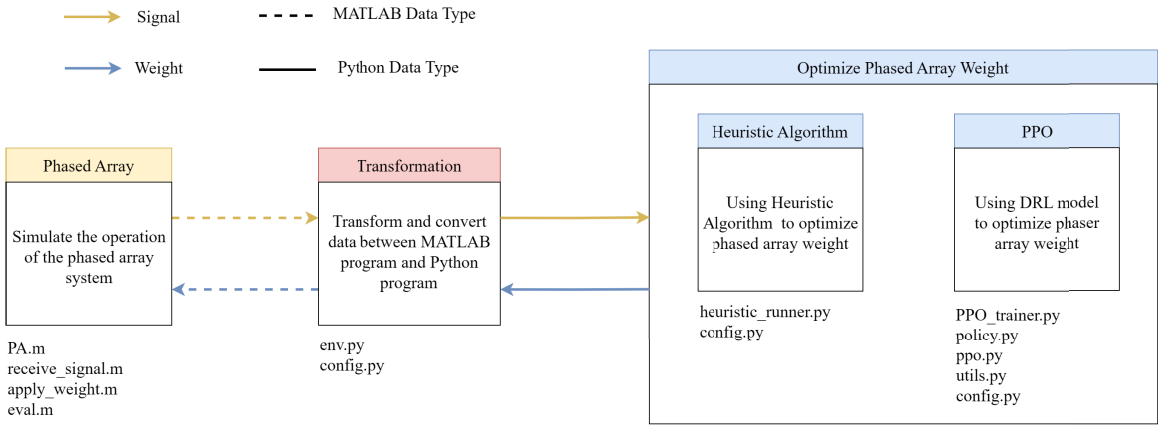

FIGURE 4. Tesdtbed.

affect DRLNuller. As the difference in received signal direction increases at each time step, indicating rapider changes in the relative position of the phased array, this may influence how DRLNuller learns from the environment. Fourth, we investigate the effect of different thresholds, η^{RL}, on the learning process of DRLNuller. Since the threshold is related to the parameter in the RL reward function, we aim to determine which threshold setting yields the best performance in our experiments.

A. EXPERIMENTAL SETUP AND PERFORMANCE METRICS

Table 4 lists the parameters used in the experiments, including the settings for the simulated signal and phased array, as well as the hyperparameters for PSO, GA, and PPO. Figure 5 illustrates the neural network architecture used for training the RL model. In the critic network, we use five fully connected layers, with the output being a value that indicates the state value. In the actor network, we employ six fully connected layers, which then split into the real and imaginary parts, since the weight of each antenna element is a complex number. The output shape of the actor depends on the number of antenna elements N^{PA} in the phased array.

The performance metrics include the average signal power in the desired direction and the interference direction. This metric indicates the ratio of energy strength between the input and output focusing in those directions, with the power values normalized and converted to units of dB. This helps us understand the performance in both the desired and interference directions. Additionally, we calculate the SIR, a common metric for measuring communication quality, to evaluate overall performance. In the comparison with the other algorithms, we also consider the calculation time (the inference time), which is the inference time to generate the phased array weights, and the convergence time (the training time), which is the time to reach the target SIR, to assess the algorithm's suitability for the environments.

B. DRLNuller VS. DDPG VS. HEURISTIC ALGORITHMS

This investigation compares SIR between DRLNuller, DDPG, PSO, and GA. Additionally, we compare the calculation and convergence times to highlight the advantages of using the DRLNuller in various scenarios.

In Figure 6, we observe that the power in the desired direction is comparable among all methods, indicating that the performance of DRLNuller and the heuristic algorithms in enhancing the desired signal is similar. However, the power in the interference direction for DRLNuller is higher than those for both PSO and GA, denoting that the heuristic algorithms possess superior interference suppression capabilities. The reason is that RL needs to explore the environment. This exploration process sometimes leads to worse performance compared to heuristic algorithms that exploit known good

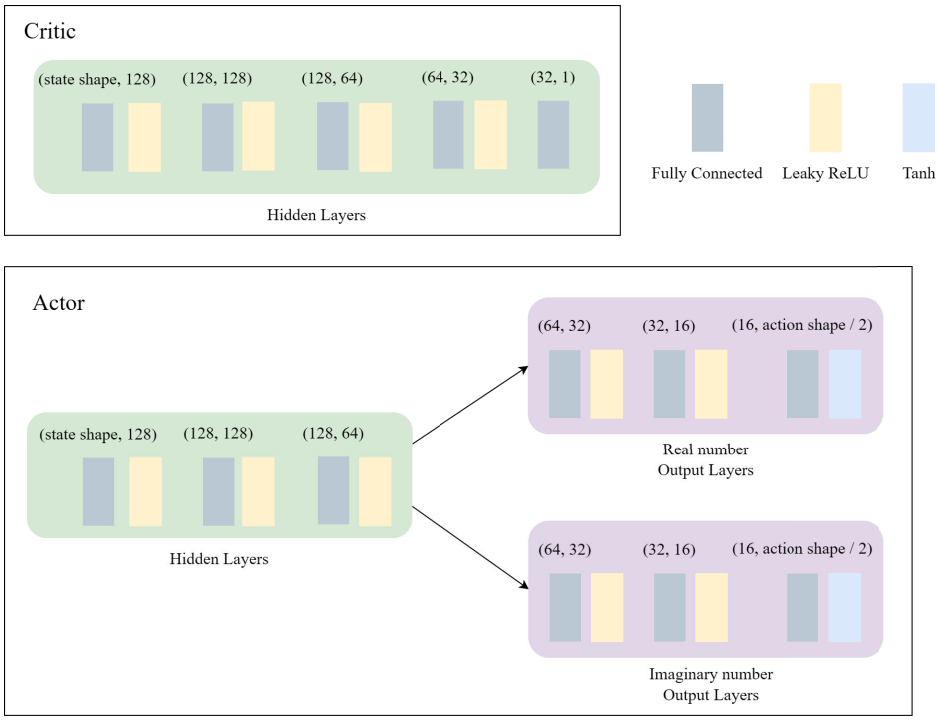

FIGURE 5. Neural network architecture.

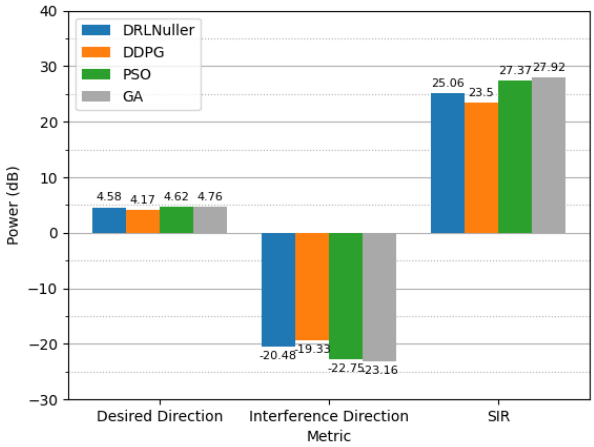

FIGURE 6. Performance on different algorithms.

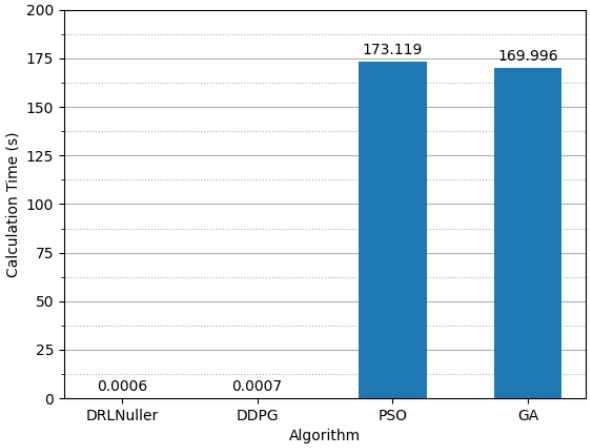

FIGURE 7. Calculation time on different algorithms.

actions. Despite this, the SIR of DRLNuller is only 9% lower than that of PSO and GA, and still achieves 25 dB, sufficiently ensuring high-quality communication in the presence of interference. In different RL algorithms, DRLNuller performs better than RL using DDPG. This is likely because DDPG is more sensitive to hyperparameters than PPO, making it harder to find suitable hyperparameter settings for training.

In Figure 7, the calculation time of DRLNuller is approximately 2.83×10^5 times faster than that of the heuristic algorithms. This is because the architecture of the actor network in DRLNuller contains only six fully connected layers, making the inference time very short. In contrast, the heuristic algorithms are iteration-based, resulting in a longer optimization process. This calculation speed is crucial in time-sensitive scenarios, as a delayed solution could prevent the phased array from effectively suppressing interference signals, thereby degrading communication quality. The calculation time between DRLNuller and DDPG is almost the same since they use identical neural network architectures for inference.

Although the calculation time of DRLNuller is significantly faster than that of the heuristic algorithms, it initially

TABLE 4. Parameter configuration for experiment.

Simulated Environment	
Phased array type	Uniform linear array
Signal's carrier frequency	100 MHz
Phased array's sampling frequency	1000 Hz
Signal's wave length	$\frac{\text{Speed of light}}{\text{Signal's carrier frequency}}$
Distance between antenna element	Signal's wave length / 2
Particle Swarm Optimization	
Terminate iteration	100
The size of swarm being used	25
The cognitive impact value, (c_1)	2.0
The social impact value, (c_2)	2.0
Genetic Algorithm	
Terminate generation	100
Population size	25
Selection function	Tournament Selection
Crossover function	Simulated Binary Crossover
Mutation function	Polynomial Mutation
Proximal Policy Optimization	
State shape	$(N^{PA})^2 + 1$
Total action shape	$2N^{PA}$
Learning rate	3×10^{-4}
Batch size	128
Rollout buffer size, (C)	2048
Discount factor	0.99
GAE lambda	0.95
Clip loss range	0.2
Value loss coefficient	0.5
Entropy loss coefficient	0.1

requires time to train its neural network. Figure 8 compares the convergence times of DRLNuller, DDPG, PSO, and GA. For DRLNuller, the time is measured in time steps, with each time step in the simulation environment defined as 1 millisecond (ms). Therefore, the convergence time is the number of time steps required for the model to reach the target SIR, multiplied by 1 ms. For the heuristic algorithms, a complete optimization process provides the solution that reaches the target SIR, so the convergence time is equivalent to the calculation time.

From Figure 8, we observe that DRLNuller requires an average of 342,786 time steps to reach the target SIR. Given that each time step is 1 millisecond, this results in a convergence time of approximately 342.786 seconds, which is about twice the time required by the heuristic algorithms. Despite the longer learning period, DRLNuller's fast calculation time still offers significant advantages in continuously changing environments. DDPG has the longest convergence time. This is because training a model using DDPG requires four neural networks, making it relatively unstable compared to the PPO algorithm which employs only two neural networks.

The long convergence time may be a bottleneck in scenarios where the environment changes abruptly and

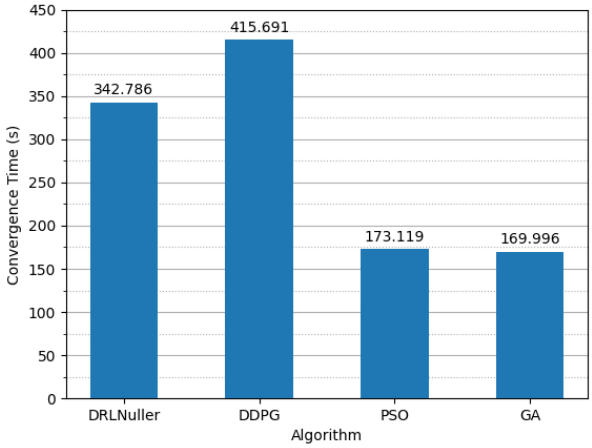

FIGURE 8. Convergence time on different algorithms.

frequently, thereby demanding frequent retraining. While our approach excels in situations with moderate changes occurring over seconds to minutes, abrupt or mission-critical shifts can indeed outpace this convergence window. Potential solutions like incremental or online RL updates that continue refining a partially trained model may substantially reduce the time needed to re-converge under sudden condition changes.

C. THE EFFECT OF PHASED ARRAY SIZE

The size of a phased array refers to the number of antenna elements N^{PA} in the phased array. In our experiments, we tested arrays with 10, 20, 30, 40 and 50 antenna elements to evaluate the effects of phased array size on the performance of DRLNuller.

In Figure 9, we conducted experiments with the same training duration to observe performance changes as the phased array size increased. The power in the interference direction increased dramatically from an average of -20.48 dB to -12.97 dB. Thus, the SIR for phased arrays with 50 antenna elements decreased by 19% compared to those with 10 antenna elements. The significant performance drop is due to the increased complexity of the solution space for optimized phased array weights, making it harder for the RL model to explore good solutions. There is a slight increase in the desired direction power. The reason is because the total power of the phased array is increased due to a higher number of antennas.

A larger phased array theoretically provides narrower beams and deeper nulls. However, our RL-based model must navigate a larger solution space as the number of elements increases, resulting in slower convergence. In our experiments, however, the training duration was kept the same for all array sizes. As the array grows, our RL-based method must navigate a much larger solution space, and the fixed-length training inevitably leads to slower convergence for bigger arrays. Consequently, Figure 9 shows a "dip"

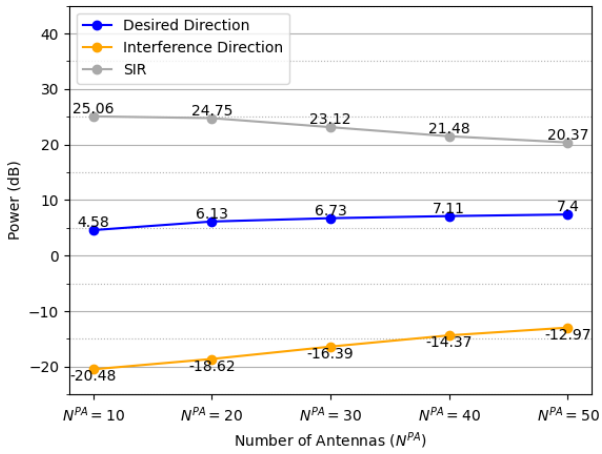

FIGURE 9. Performance on different phased array size.

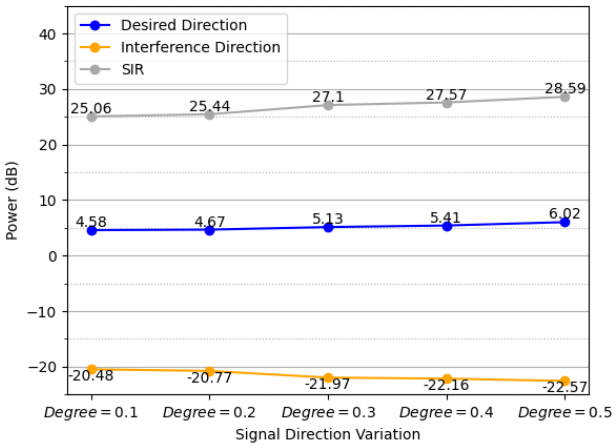

FIGURE 10. Performance on different signal direction variation.

in performance for the largest array because it had not yet reached its optimal solution within the allotted fixed duration.

With additional training iterations or more extensive hyperparameter tuning, the suppression performance for larger arrays can be further improved, ultimately realizing the expected advantages of a larger phased array size and being consistent with theoretical expectations.

D. THE EFFECT OF SIGNAL DIRECTION VARIATION

We define signal direction variation as the incremental angular change between consecutive time steps. For example, if the variation is set to 0.3°, then a signal arriving at 30° at time step t will arrive at 30.3° at time step t + 1. To examine how direction variation affects DRLNuller's performance, we tested five settings 0.1°, 0.2°, 0.3°, 0.4°, and 0.5° and evaluated the results under each scenario.

In Figure 10, we observe that as the signal direction variation increases, the power in the desired direction rises. The power in the interference direction also decreases. As a results, SIR rised by approximately 14%, from 25.06 dB to 28.59 dB, meeting the requirements of scenarios demanding high communication quality. We believe the reason for this performance improvement is that the differences between the two received signals become more distinct, making it easier for the neural network model to differentiate them. As a result, the model can produce a better adjustment.

E. THE EFFECT OF INTERFERENCE SIGNAL THRESHOLD

We also tested different constraint conditions by modifying the interference signal threshold η^{RL}. If the model's action evaluation did not meet the constraint, the reward R_t^{RL} used for updating the model was set directly to a negative value. We experimented with five constraint settings, 0, −5, −10, −15 and −20, and observed how stricter constraints affect the model's learning to determine which configuration provides the best performance of DRLNuller.

In Figure 11, we observe that as the constraints become stricter, the model appears to concentrate on improving the power in the interference direction, although the power in the desired direction also becomes lower. This phenomenon could be due to the strict constraint only accounting for the interference power, making the number of violations of the constraint on interference direction greater. As a result, the model focuses more on not violating the constraint on interference direction. The overall SIR decreases by approximately 5% compared to the lenient constraint within the same training duration. This occurs because, in the early stages of training, the model incurs more penalty rewards for violating constraints, requiring more time to learn better actions.

F. INTERPRETATION OF DRLNuller PERFORMANCE

A holistic inspection of Figure 6 to 11 reveals several consistent behavioral patterns of the DRLNuller and is summarized as follows.

Exploration–exploitation balance: While DRLNuller's instantaneous interference suppression is slightly weaker than that of PSO and GA because the model must occasionally explore sub-optimal actions, its policy converges to a stable 25 dB SIR and delivers an inference latency that is 2.83×10^5 times faster—decisive in real-time deployments. The short inference time offsets the exploratory penalties observed in early episodes, explaining why overall communication quality is comparable to heuristic baselines.

Sensitivity to solution-space dimensionality: When the number of antenna elements grows from 10 to 50, the achievable SIR temporarily dips by about 19%. The model must search a larger continuous weight manifold, and the fixed training budget therefore becomes the dominant performance limiter rather than the policy architecture itself. Extending training time or employing curriculum learning is expected to recover the theoretical beam-forming gain of larger arrays.

Advantage of rapid directional discrimination: Increasing the per-step signal-direction variation from 0.1° to 0.5° makes the desired and interference signatures more orthogonal;

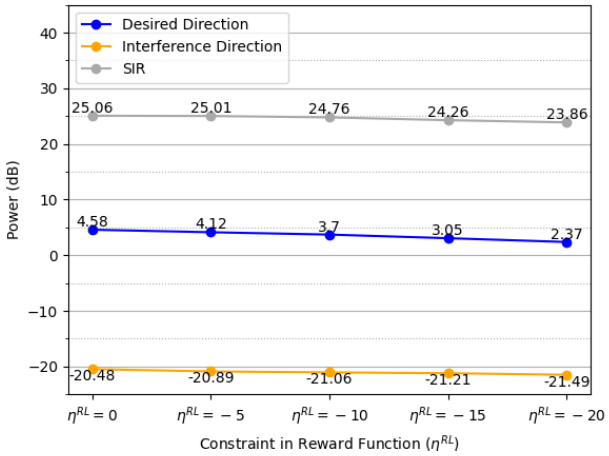

FIGURE 11. Performance on strict constraint.

consequently the model separates them more easily and the SIR rises by 14%. This trend indicates that DRLNuller leverages richer temporal diversity to assign credit an attribute that can be exploited in highly dynamic airborne or vehicular links.

Reward-driven shift: Tightening the interference-power threshold compels the model to prioritize suppressing interference, even at the cost of desired-signal gain, leading to a modest 5% SIR loss under the strictest setting. The behavior confirms that the reward cleanly encodes operator preferences: harsher penalties cause the policy to trade main-lobe gain for deeper nulls.

Collectively, these observations substantiate that DRLNuller adapts its policy in a principled manner to disparate operational regimes, it flavors speed where latency dominates, accepts slower convergence in high-dimensional spaces, capitalizes on naturally decor-related trajectories, and follows reward-imposed priorities. These insights can guide future enhancements such as adaptive training budgets, dynamic reward re-weighting, and hybrid on-policy/off-policy fine-tuning.

VII. CONCLUSION

This work proposes an innovative approach, DRLNuller, for adaptive nulling in a phased array using DRL. DRLNuller dynamically adjusts phased array weights in response to changing interference and signal conditions in a dynamic environment. The results demonstrate that DRLNuller can effectively optimize nulling performance with a considerable reduction in the calculation time compared to traditional heuristic methods.

Our experimental results show that DRLNuller outperforms PSO and GA in several areas. Notably, DRLNuller achieves a calculation speed of 2.83×10^5 times faster than these heuristic methods, making it highly suitable for real-time applications where rapid response is crucial. Although there is a minor reduction in SIR compared to heuristic algorithms, DRLNuller still ensures high communication quality with an average SIR of around 25.06 dB. Furthermore, the model's performance was tested across different phased array sizes and varying signal direction changes. While the complexity increases with a larger number of array elements, DRLNuller maintains effective communication quality, although with some decreases in SIR. Increasing the signal direction variation improves the model's ability to distinguish between desired and interference signals, leading to better performance. On the other hand, compared with another RL algorithm using DDPG, DRLNuller has a higher SIR, lower calculation time, and convergence time, demonstrating that DRLNuller adopts a better RL algorithm, PPO.

In the future, we can explore several promising directions. First, testing with multiple interference sources could yield deeper insights. Although our current experiments focus on a single interference for simplicity, real-world deployments often involve multiple interferences at various angles and power levels. Thus extending DRLNuller to conquer with an environment which involves multiple dynamic interferers is worth studying. Practical deployment must also cope with hardware realities. A calibration-robust version of DRLNuller can be trained by injecting noise into the DRLNuller's output during simulation. Another potential avenue is designing hybrid models that combine the strengths of DRL and heuristic algorithms to achieve more efficient optimization. Such approaches could harness the rapid inference capabilities of DRL alongside the robust search strategies of heuristic methods.

REFERENCES

[1] C.-H. Hsu, C.-H. Chen, W.-J. Shyr, K.-H. Kuo, Y.-N. Chung, and T.-C. Lin, "Optimizing beam pattern of linear adaptive phase array antenna based on particle swarm optimization," in *Proc. 4th Int. Conf. Genetic Evol. Comput.*, Dec. 2010, pp. 586–589.

[2] L. A. Greda, A. Winterstein, D. L. Lemes, and M. V. T. Heckler, "Beam-steering and beamshaping using a linear antenna array based on particle swarm optimization," *IEEE Access*, vol. 7, pp. 141562–141573, 2019.

[3] M. E. Yigit and T. Gunel, "Pattern synthesis of linear antenna array via a new hybrid Taguchi–genetic–particle swarm optimization algorithm," in *Proc. 18th Medit. Microw. Symp. (MMS)*, Oct. 2018, pp. 17–21.

[4] A. Rahmanti, I. W. Mustika, and Selo, "Moth flame optimization for weight adjustment on phased array antenna," in *Proc. Int. Conf. Inf. Technol. Syst. Innov. (ICITSI)*, Nov. 2022, pp. 118–121.

[5] S. Lu, S. Zhao, and Q. Shi, "Learning-based massive beamforming," in *Proc. IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.

[6] S. Bianco, P. Napoletano, A. Raimondi, M. Feo, G. Petraglia, and P. Vinetti, "AESA adaptive beamforming using deep learning," in *Proc. IEEE Radar Conf. (RadarConf20)*, Sep. 2020, pp. 1–6.

[7] F. Zardi, P. Nayeri, P. Rocca, and R. Haupt, "Artificial intelligence for adaptive and reconfigurable antenna arrays: A review," *IEEE Antennas Propag. Mag.*, vol. 63, no. 3, pp. 28–38, Jun. 2021.

[8] R. Lovato and X. Gong, "Phased antenna array beamforming using convolutional neural networks," in *Proc. IEEE Int. Symp. Antennas Propag. USNC-URSI Radio Sci. Meeting*, Jul. 2019, pp. 1247–1248.

[9] A. H. Sallomi and S. Ahmed, "Multi-layer feed forward neural network application in adaptive beamforming of smart antenna system," in *Proc. Al-Sadeq Int. Conf. Multidisciplinary IT Commun. Sci. Appl. (AIC-MITCSA)*, May 2016, pp. 1–6.

[10] P. Ramezanpour and M.-R. Mosavi, "Two-stage beamforming for rejecting interferences using deep neural networks," *IEEE Syst. J.*, vol. 15, no. 3, pp. 4439–4447, Sep. 2021.

[11] X. Xiao and Y. Lu, "Data-based model for wide nulling problem in adaptive digital beamforming antenna array," *IEEE Antennas Wireless Propag. Lett.*, vol. 18, pp. 2249–2253, 2019.
[12] T. Sallam and A. M. Attiya, "Convolutional neural network for 2D adaptive beamforming of phased array antennas with robustness to array imperfections," *Int. J. Microw. Wireless Technol.*, vol. 13, no. 10, pp. 1096–1102, Dec. 2021.
[13] J. Wang, F. Wang, B. Ding, W. Luo, and S. Niu, "A personnel-free method for array synthesis based on artificial intelligence," in *Proc. IEEE 10th Asia–Pacific Conf. Antennas Propag. (APCAP)*, Nov. 2022, pp. 1–2.
[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[15] L. M. R. Rere, M. I. Fanany, and A. M. Arymurthy, "Simulated annealing algorithm for deep learning," *Proc. Comput. Sci.*, vol. 72, pp. 137–144, Jan. 2015.
[16] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, vol. 4, 2002, pp. 1942–1948.
[17] M. Mitchell, *An Introduction To Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.
[18] S. Fujimoto, H. V. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.
[19] K.-B. Yu and M. F. Fernandez, "Methods to combine deterministic nulling and adaptive nulling," in *Proc. IEEE Radar Conf. (RadarConf)*, May 2017, pp. 0123–0128.
[20] I. Mallioras, Z. D. Zaharis, P. I. Lazaridis, and S. Pantelopoulos, "A novel realistic approach of adaptive beamforming based on deep neural networks," *IEEE Trans. Antennas Propag.*, vol. 70, no. 10, pp. 8833–8848, Oct. 2022.
[21] I. Mallioras, Z. D. Zaharis, P. I. Lazaridis, V. Poulkov, N. V. Kantartzis, and T. V. Yioultsis, "An adaptive beamforming approach applied to planar antenna arrays using neural networks," in *Proc. IEEE Int. Black Sea Conf. Commun. Netw. (BlackSeaCom)*, Jun. 2022, pp. 293–297.
[22] Y. Zhang, T. Osman, and A. Alkhateeb, "Online beam learning with interference nulling for millimeter wave MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 23, no. 5, pp. 5109–5124, May 2024.
[23] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.
[24] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2623–2631.
[25] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in Python," *IEEE Access*, vol. 8, pp. 89497–89509, 2020.
[26] M. Waskom, "Seaborn: Statistical data visualization," *J. Open Source Softw.*, vol. 6, no. 60, p. 3021, Apr. 2021.
[27] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007.

YING-DAR LIN (Fellow, IEEE) received the Ph.D. degree in computer science from the University of California at Los Angeles (UCLA), in 1993. He was a Visiting Scholar with Cisco Systems, San Jose, from 2007 to 2008, the CEO of the Telecom Technology Center, Taiwan, from 2010 to 2011, and the Vice President of National Applied Research Labs (NARLabs), Taiwan, from 2017 to 2018. He co-founded L7 Networks Inc., in 2002, and O'Prueba Inc., in 2018. He is currently a Chair Professor of computer science with National Yang Ming Chiao Tung University (NYCU), Taiwan. His research interests include cybersecurity, wireless communications, network softwarization, and machine learning for communications. He served or is serving on the editorial boards for several IEEE journals and magazines, including the Editor-in-Chief for IEEE COMMUNICATIONS SURVEYS AND TUTORIALS (2017–2020).

JEN-HAO CHANG received the M.S. degree from the Institute of Network Engineering, National Yang Ming Chiao Tung University (NYCU), in 2024. He was an Associate Researcher with the High Speed Network Laboratory, NYCU, from 2022 to 2024. His research interests include control problem and deep learning.

YUAN-CHENG LAI (Member, IEEE) received the Ph.D. degree from the Department of Computer and Information Science, National Chiao Tung University, in 1997. He joined the Department of Information Management, National Taiwan University of Science and Technology, in August 2001, and has been a Distinguished Professor, since June 2012. His research interests include performance analysis, software-defined networking, wireless networks, and the IoT security.