Harnessing Monotonic Neural Networks for Performance Prediction and Threshold Determination in Multichannel Detection

Rui Zhou, *Member, IEEE*, Wenqiang Pu, Ming-Yi You, and Qingjiang Shi, *Member, IEEE*

***Abstract*—Despite extensive research on numerous multichannel detection methods, predicting their performance remains difficult due to the high dimensionality of raw data and the complexity of the detection process. To tackle this, we introduce a special type of neural network designed to predict detection performance under specific environmental conditions. We utilize a monotonic neural network (MNN) to develop PdMonoNet, which ensures that the influence of input parameters on the output probability of detection is monotonic. This approach also facilitates the determination of thresholds. We provide a theoretical analysis of the universal approximation capabilities and prediction error of the network architectures we employ. Numerical experiments conducted on both synthetic datasets and real-world scenarios within the context of multichannel spectrum sensing demonstrate the effectiveness and robustness of PdMonoNet in predicting detection performance and determining thresholds.**

***Index Terms*—Multichannel detection, performance prediction, threshold determination, monotonic neural network.**

I. Introduction

MULTICHANNEL detection has emerged as a pivotal technique across a range of disciplines, including communications, radar, acoustics, and medicine [1], [2], [3], [4]. This method utilizes arrays of receivers or multiple spatially distributed receivers to detect the presence of target signals. In communications, for instance, multichannel detection is crucial for cognitive radio, allowing secondary users to identify unoccupied licensed frequency bands [5], [6], [7], [8], [9], [10]. In radar technology, it enhances detection performance within distributed radar networks and passive radar systems [2], [11]. In the field of acoustics, it is essential for voice activity detection, helping to differentiate between periods of speech and silence in audio streams [3], [12], [13]. In medical applications, multichannel detection facilitates the diagnosis, monitoring, and treatment of various disorders, including epilepsy, sleep apnea, and muscular diseases [4], [14], [15]. The versatility and increasing applications of multichannel detection techniques make them a significant and increasingly researched topic within these fields.

Multichannel detection methods have progressively evolved, significantly enhancing detection accuracy and reliability while increasing in sophistication and complexity. The most basic form, energy detection, measures the cumulative energy level of signals across all receivers and compares it to a predefined threshold to determine the presence of target signals [16]. A refined version adapted for spatially distributed receivers requires each receiver to transmit its binary decision to a fusion center, thereby reducing communication costs and improving the robustness of detection outcomes. Standard detection approaches include Logical-OR, Logical-AND, and Majority rules [6], [17]. Energy detection, however, tends to perform poorly under low signal-to-noise ratio (SNR) conditions and is susceptible to noise uncertainty [18], [19]. Instead, researchers have developed more advanced techniques that utilize the correlation structure of received data. For example, the eigenvalue arithmetic-to-geometric mean (AGM) detector computes a statistic from the ratio of the arithmetic to geometric means of eigenvalues, using the dispersion of the eigenspectrum as an indicator of a primary signal's presence [20]. Other advanced methods include the eigenvalue-moment-ratio (EMR) detector [21], the maximum-minimum eigenvalue (MME) detector [22], the scaled largest eigenvalue (SLE) detector [23], and the generalized likelihood ratio test (GLRT) detector [24]. The recent integration of machine learning methods in multichannel detection has enabled the identification of complex and dynamic signal patterns, drastically improving detection capabilities. Techniques such as support vector machines [25], random forests [26], and convolutional neural networks [4], [15], [27] have demonstrated enhanced accuracy and adaptability across diverse scenarios and SNR conditions. As technological advancements continue, the development of increasingly intricate multichannel detection methods remains a vibrant and dynamic field of research [13], [28], [29].

Received 18 July 2024; revised 24 December 2024 and 14 April 2025; accepted 30 April 2025. Date of publication 8 May 2025; date of current version 13 June 2025. This work was supported in part by the National Nature Science Foundation of China (NSFC) under Grant 62201362 and Grant 62101350, in part by Shenzhen Science and Technology Program under Grant RCBS20221008093126071, and in part by the Starlit South Lake Leading Elite Program. The associate editor coordinating the review of this article and approving it for publication was Kobi Cohen. *(Corresponding authors: Wenqiang Pu; Ming-Yi You.)*

Rui Zhou and Wenqiang Pu are with Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong-Shenzhen, Guangdong 5181722, China (e-mail: rui.zhou@sribd.cn; wpu@sribd.cn).

Ming-Yi You is with the National Key Laboratory of Electromagnetic Space Security, Jiaxing 3140333, China (e-mail: youmingyi@126.com).

Qingjiang Shi is with the School of Software Engineering, Tongji University, Shanghai 200092, China, and also with Shenzhen Research Institute of Big Data, Shenzhen 518172, China (e-mail: shiqj@tongji.edu.cn).

Digital Object Identifier 10.1109/TSP.2025.3567761

While extensive research has focused on developing sophisticated multichannel detection methods, the theoretical foundations for measuring their detection performance have lagged behind [25], [30], [31], [32]. Theoretical analyses of even basic detection methods often rely on ideal conditions, overlooking practical variables that significantly affect performance in real-world scenarios [33], [34]. For example, standard energy detection assumes perfect knowledge of noise power and a large sample size [30], which are rarely met in practice. It may result in suboptimal thresholds and unreliable detection performance [33]. The absence of a feasible measurement framework presents a greater challenge for evaluating complex methods, such as those involving machine learning and deep learning, particularly in diverse working environments [25], [31], [32]. Without adequate measurement tools, optimizing the parameter setting of advanced detection techniques, like deploying the receivers or adjusting directional antennas' orientations, is challenging and often confined to simpler methods like energy detection [35]. Therefore, Monte Carlo simulation has served as a popular and convenient method for assessing the performance of multichannel detection methods [22], [24], [36]. This technique generates a vast amount of random data for the scenario under study and evaluates detection performance empirically. However, these simulations are computationally intensive and require numerous iterations to achieve statistically significant results, particularly for complex detection methods and extensive networks. Furthermore, Monte Carlo simulations function as a black-box approach, providing no mathematical relationships that govern detection performance [37]. This opacity restricts deeper insights into optimal deployment and settings for multichannel detection methods.

Consequently, there is a pressing need to develop models or learning approaches that can efficiently and accurately gauge the performance of multichannel detection methods. Recent research increasingly applies machine learning to predict the performance of methods in various fields, such as positioning uncertainty [38], [39] and the Cramér-Rao bound [40], as well as to estimate parameters [41]. These studies have demonstrated remarkable accuracy and efficiency. Accordingly, the primary objective of this paper is to predict the probability of detection and determine thresholds in multichannel detection using neural networks, with spectrum sensing serving as a case study. The main contributions of this paper are outlined as follows:

- We first examine the challenges of the prevalent use of Monte Carlo simulations for assessing detection performance and thresholds. A theoretical analysis of the the required number of Monte Carlo simulations for achieving a desired reliability level is presented. Our findings suggest that an extensive number of simulations are necessary to obtain a precise estimate of detection performance, particularly when the probability of false alarms is low.
- To effectively predict detection performance, we initially introduce PdNet, a preliminary method that utilizes a classical multilayer perceptron (MLP) neural network. However, the MLP architecture cannot directly accommodate the inherent monotonic relationships between system parameters and detection performance. To address this limitation, we propose PdMonoNet, a novel approach that employs a monotonic neural network (MNN). Additionally, we investigate the use of both MLP and MNN models in determining the necessary thresholds for multichannel detection, which we refer to as ThreshNet and ThreshMonoNet, respectively.
- In theoretical terms, we discuss the universal approximation capabilities of our proposed PdMonoNet approach and its prediction accuracy concerning unseen data points. The results indicate that PdMonoNet can universally approximate the desired detection performance function under mild conditions, with bounded prediction errors at unseen data points.
- Numerical experiments on both synthetic and real-world datasets are conducted to evaluate our proposed methods for performance prediction and threshold determination in multichannel detection. Our results demonstrate consistent monotonic behavior and robustness against unseen data points, even in scenarios involving outliers.

This paper is structured as follows. Sec. II introduces the system model for multichannel detection and discusses the challenges associated with predicting the detection performance. Sec. III describes the classical Monte Carlo simulation technique and introduces the naive PdNet approach. In Sec. V, we introduce PdMonoNet, which utilizes a MNN to measure the probability of detection. This section also includes a theoretical analysis of the PdMonoNet's universal approximation properties and its prediction accuracy for unseen data points. The approach for determining decision thresholds using the MNN is detailed in Sec. V. Sec. VI provides numerical experiments to evaluate the performance of the proposed prediction and determination methods. Finally, Sec. VII summarizes the conclusions of the study.

In this paper, we use lowercase letters (e.g., x) to denote scalars, bold lowercase letters (e.g., $\mathbf{x}$) to denote vectors, and bold uppercase letters (e.g., $\mathbf{X}$) to denote matrices. The set of real numbers is represented by $\mathbb{R}$, and the set of complex numbers by $\mathbb{C}$. The Frobenius norm and p-norm of a matrix are denoted by $\|\mathbf{X}\|_F$ and $\|\mathbf{X}\|_p$, respectively, and the transpose of a matrix is represented by $\mathbf{X}^T$. The p-norm of a vector $\mathbf{x}$ is denoted by $\|\mathbf{x}\|_p$. The partial derivative is represented by $\partial(\cdot)$, the indicator function by $\mathbb{1}(\cdot)$, and the absolute value of a scalar by $|\cdot|$.

II. System Model

In this section, we briefly introduce the system model for multichannel detection and subsequently discuss the challenges in predicting the detection performance of detection methods.

A. Multichannel Detection

Multichannel detection is a widely utilized technique in various disciplines, including communications, radar, acoustics, and medicine. Its primary objective is to detect the presence of a target signal through the collaboration of multiple receivers.

The problem of multichannel detection is typically formed as a classical hypothesis testing problem, defined as follows:

$$\mathcal{H}_0 : \mathbf{x}_i = \mathbf{n}_i, \quad \mathcal{H}_1 : \mathbf{x}_i = \mathbf{s}_i + \mathbf{n}_i, \tag{1}$$

where $\mathbf{x}_i \in \mathbb{C}^L$ is the received signals of L receivers at time index i, $\mathbf{n}_i \in \mathbb{C}^L$ is noise at time index i, $\mathbf{s}_i \in \mathbb{C}^L$ signifies the target signal arriving at L receivers at time index i. The core task is to determine whether the target signal $\mathbf{s}$ is present in the received signal $\mathbf{x}$.

Various methods have been developed to address Problem (1), typically assuming that the signal is uncorrelated with Gaussian noise [21], [22], [24]. In general, these methods obey the following decision rule:

$$\xi = T(\mathbf{X}) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{>}} rless\ \gamma, \tag{2}$$

where $T(\cdot)$ is a function that calculates the test statistic ξ from N received samples, collected as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{C}^{L\times N}$, and the γ is a predefined threshold to determine the presence of a target signal. More specifically, the target signal is considered present if $T(\mathbf{X}) \geq \gamma$, and absent if $T(\mathbf{X}) < \gamma$. The energy detector compute the test statistics simply as $T(\mathbf{X}) = \|\mathbf{X}\|_F^2$ [30]. The machine learning-based detectors employ neural networks to serve as the test statistic function $T(\mathbf{X})$ [25], [31], [32], introducing a more sophisticated approach. This remains an active area of research with continuous development of novel detection methods.

B. Difficulties of Predicting Detection Performance

Once function $T(\cdot)$ has been determined, a pertinent question arises regarding its performance. Specifically, assuming the presence of a target signal, and with all environmental parameters such as sample rate, the signal-to-noise ratio (SNR) of receivers, etc, represented as $\boldsymbol{\theta}$, we aim to obtain the probability of detection function, denoted as $\ell(\boldsymbol{\theta})$. The typical approach involves modeling the theoretical distribution of the obtained test statistics, $p(\xi \mid \boldsymbol{\theta})$, and generally consists of two steps:

1) **Threshold Determination**: Obtain the threshold γ by setting the desired probability of false alarm (P_{FA}). This is calculated as:

$$\gamma = \arg\min_{\xi} \left\{ \int_{\xi}^{\infty} p(\tilde{\xi} \mid \tilde{\boldsymbol{\theta}}) d\tilde{\xi} \leq P_{FA} \right\}, \tag{3}$$

where $\tilde{\boldsymbol{\theta}}$ denotes an environment parameter that similar to $\boldsymbol{\theta}$ but with the primary signal absent at the receivers.

2) **Detection Probability Prediction**: Predict the probability of detection as:

$$\ell(\boldsymbol{\theta}) = \int_{\gamma}^{\infty} p(\xi \mid \boldsymbol{\theta}) d\xi. \tag{4}$$

However, obtaining the distribution of ξ is a challenging task. Specifically, the computation of $p(\xi \mid \boldsymbol{\theta})$ is mathematically expressed as:

$$p(\xi = \xi_0 \mid \boldsymbol{\theta}) = \int \mathbb{1}\left[T(\mathbf{X}) = \xi_0\right] p(\mathbf{X} \mid \boldsymbol{\theta})\, d\mathbf{X}, \tag{5}$$

where $\mathbb{1}(\cdot)$ is the indicator function, $p(\mathbf{X} \mid \boldsymbol{\theta})$ is the conditional distribution of $\mathbf{X}$ given $\boldsymbol{\theta}$. The difficulties in obtaining this distribution are considered intractable for the following reasons:

1) **High-dimensionality of $\mathbf{X}$**: The dimensionality of $\mathbf{X}$ is usually very larger. This aspect complicates the integration detailed in (5), making it challenging to calculate.
2) **Complexity of $T(\cdot)$**: Many detection methods involve complex data processing within $T(\cdot)$. For instance, the GLRT detector [24] requires solving constrained non-convex maximum likelihood estimation problems for each evaluation of $T(\cdot)$, while deep learning approaches involve multiple layers of data processing. This complexity makes it nearly impossible to precisely identify the set of $\mathbf{X}$ that results in $T(\mathbf{X}) = \xi_0$.

According to the above discussions, we see that deriving an exact function to predict detection performance, $\ell(\boldsymbol{\theta})$, is an intractable challenge. There is limited existing research that addresses this issue. The few studies that do exist are typically limited to detectors employing simple transformations, such as $T(\mathbf{X}) = \|\mathbf{X}\|_F^2$ in energy detectors when N is sufficiently large [30]. Alternatively, some studies approximate detection performance using advanced statistical tools, such as random matrix theory (RMT) [21], [30]. But these approximations are typically tailored to specific detection methods and necessitate considerable additional effort.

Understanding $\ell(\boldsymbol{\theta})$ is crucial not only for evaluating the performance of existing detectors but also for advancing research in areas like optimizing receiver deployment to maximize detection capabilities. In the subsequent section, we will first revisit the naive approach of estimating $\ell(\boldsymbol{\theta})$ using the Monte Carlo method. Then we will introduce our preliminary approach using neural networks to approximate this function. The ensuing discussion will focus on the application of these methods in spectrum sensing as a case study.

III. The Preliminary Approaches

A. Preliminary Approach I: Monte Carlo Simulation

Rather than deriving a mathematical expression for detecting performance $\ell(\boldsymbol{\theta})$, the majority of the literature relies on Monte Carlo experiments to empirically simulate the performance of detectors of interest. This approach typically involves two steps:

1) **Threshold Approximation**: Generate K realizations of received pure noise $\{\mathbf{X}_k\}_{k=1}^K$ under $\tilde{\boldsymbol{\theta}}$. Estimate the threshold $\hat{\gamma}$ by applying the desired false alarm probability P_{FA}, calculated as:

$$\hat{\gamma} = \arg\min_{x} \left\{ \frac{\sum_{k=1}^K \mathbb{1}\left(T(\mathbf{X}_k) \geq x\right)}{K} \leq P_{FA} \right\}, \tag{6}$$

2) **Detection Probability Evaluation**: Generate M realizations of received signals $\{\mathbf{X}_m\}_{m=1}^M$ using the environment setting $\boldsymbol{\theta}$, and then evaluate $\ell(\boldsymbol{\theta})$ as:

$$\hat{\ell}(\boldsymbol{\theta}) = \frac{\sum_{m=1}^M \mathbb{1}\left(T(\mathbf{X}_i) \geq \hat{\gamma}\right)}{M}. \tag{7}$$

Though direct and convenient, the Monte Carlo simulation method requires sufficiently large K and M to produce reliable results, as detailed in the following facts: Fact 1 and Fact 2.

Fact 1: Given $\hat{\gamma}$ obtained from (6) using K realizations to match a target false alarm probability P_{FA}, the 95% confidence interval for $P_{FA}(\hat{\gamma})$ can be approximated as follows:

$$P_{FA} \pm 1.96 \times \sqrt{\frac{\tilde{P}_{FA}(1-\tilde{P}_{FA})}{K}}, \tag{8}$$

where $\tilde{P}_{FA} = (P_{FA} \times K + 0.5 \times 1.96^2)/(K + 1.96^2)$.

Proof: Consider the empirical estimate of the false alarm rate, denoted by $\hat{P}_{FA} = \frac{x}{K}$, where x follows a binomial distribution, $x \sim \text{Binomial}(K, P_{FA}(\hat{\gamma}))$. Given that the threshold $\hat{\gamma}$ is chosen such that $\hat{P}_{FA} = P_{FA}$, the confidence interval for $P_{FA}(\hat{\gamma})$ can be determined using the Agresti–Coull method [42]. ■

Fact 2: Given $\hat{\ell}(\boldsymbol{\theta})$ obtained from (7) using M realizations, the expected absolute difference from the true $\ell(\boldsymbol{\theta})$ can be expressed as follows:

$$\mathbb{E}\left[\left|\hat{\ell}(\boldsymbol{\theta}) - \ell(\boldsymbol{\theta})\right|\right] \approx \sqrt{\frac{2}{\pi}}\sqrt{\frac{\ell(\boldsymbol{\theta})(1-\ell(\boldsymbol{\theta}))}{M}}. \tag{9}$$

Proof: Consider the empirical estimate of $\ell(\boldsymbol{\theta})$ given by $\hat{\ell}(\boldsymbol{\theta}) = \frac{x}{M}$, where x follows a binomial distribution, $x \sim \text{Binomial}(M, \ell(\boldsymbol{\theta}))$. As M increases, the distribution of $\hat{\ell}(\boldsymbol{\theta})$ approximates a normal distribution, $\mathcal{N}(\ell(\boldsymbol{\theta}), \frac{\ell(\boldsymbol{\theta})(1-\ell(\boldsymbol{\theta}))}{M})$, in accordance with the central limit theorem [43]. This approximation leads to the formulation in (9). ■

For practical application, to accurately estimate a low P_{FA} such as 1%, a significantly large K is necessary to narrow the confidence interval around P_{FA}. For instance, aiming for a margin of error of 0.2% around P_{FA} requires $K \approx 10,000$. Similarly, even fed with an accurate threshold γ, to ensure that $\ell(\boldsymbol{\theta})$ is estimated within a margin of error of 0.02 at $\ell(\boldsymbol{\theta}) = 0.5$, $M \approx 400$ is needed. We present a straightforward example in Fig. 1, which uses the same parameter settings as those in Fig. 4(a). It is evident that the probability of detection results are prone to fluctuation when $K = 1,000$ and $M = 100$. Even when K and M are increased to $10,000$ and $1,000$, respectively, fluctuations remain observable. The only scenario where a smooth curve is observed is when $K = 100,000$ and $M = 10,000$.

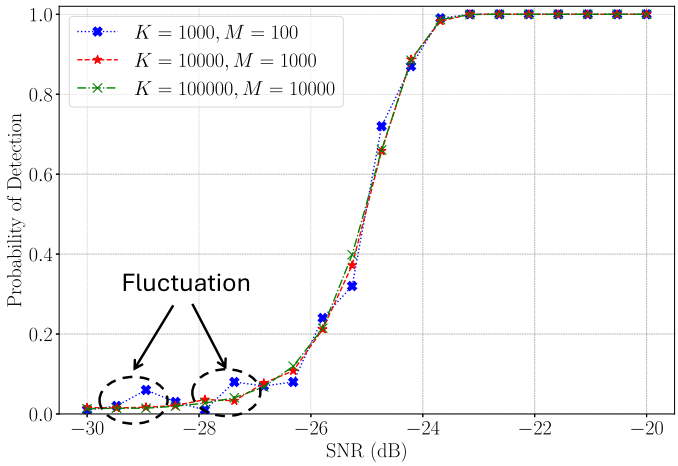


Fig. 1. An example of the probability of detection obtained through Monte Carlo experiments.

B. Preliminary Approach II: MLP

In recent years, neural networks have revolutionized the field of predictive analytics due to their ability to model complex and nonlinear relationships within data. Recent research increasingly applies machine learning to predict the performance of methods in various fields, such as positioning uncertainty [38], [39] and the Cramér-Rao bound [40], as well as to estimate parameters [41].

In response to these advancements, an MLP neural network can be directly applied to predict detection performance. The methodology involves several key steps. For example, a dataset $\mathcal{D}$ as outlined in Apx. A is generated. Subsequently, an MLP consists of an input layer, multiple hidden layers, and an output layer, all fully connected are employed to fit the generated dataset $\mathcal{D}$. Each layer sequence comprises a linear transformation followed by batch normalization and a ReLU activation function. The selection of the ReLU activation function is due to its computational simplicity and empirical effectiveness, which have been well-documented across various tasks [44]. The multi-layer structure allows it to effectively learn nonlinear functions. For simplicity, we refer to this method as PdNet. Specifically, PdNet is designed to transform an input feature space characterized by a 14-dimensional vector $\boldsymbol{\theta}$[1] into a scalar output.

C. Limitations of Preliminary Approaches

Both Monte Carlo simulation and PdNet methods exhibit limitations that may hinder their practical application in real-world scenarios.

For Monte Carlo simulation method, it necessitates a sufficiently large number of simulations (denoted as K and M) to achieve reliable results. The synthetic generation of signal $\mathbf{X}$ and evaluation of $T(\mathbf{X})$ require significant computational resources, making the repeated execution of these procedures time-consuming. Additionally, the process of deriving Monte Carlo results is often treated as a black box, any new environment parameter requires independent new simulations.

In contrast, using MLP circumvents the need for repetitive experiments and leverages auto-gradient tools to access its gradients. Nonetheless, it inherits several limitations from traditional neural network architectures:

1) **Monotonicity**: MLP models often fail to ensure the monotonicity of their outputs. For example, the literature consistently shows that detection probability should not decrease as SNR increases, assuming all other parameters

[1]As detailed in Apx. A, here $\boldsymbol{\theta} \in \mathbb{R}^{14}$ comprises the baud rate of the primary signal, sample rate, probability of false alarm, observation time, and the SNR for up to 10 nodes.

TABLE I
THE EMPIRICAL PARTIAL CORRELATION IN $\mathcal{D}$

Feature	Notation	Coeff.	95% CI	p-value
Baud Rate	f_b	0.04	[0.04, 0.04]	< 0.001
Sample Rate	f_s	0.18	[0.18, 0.18]	< 0.001
Prob. of False Alarm	P_{FA}	0.23	[0.23, 0.23]	< 0.001
Observation Time	t	0.03	[0.03, 0.03]	< 0.001
SNR (first node)	SNR	0.15	[0.15, 0.15]	< 0.001

are constant. However, traditional MLP models do not guarantee this monotonicity.

2) **Generalization Ability**: The performance of our proposed PdNet heavily relies on the training dataset and tends to degrade with unseen data.
3) **Sensitivity to Outliers**: MLP generally exhibits a lack of robustness to outliers.

In the following section, we introduce a novel method named PdMonoNet, which is based on monotonic neural networks. This approach is expected to outperform PdNet in these areas.

IV. PDMONONET

In this section, we introduce the PdMonoNet, a novel approach for predicting the probability of detection that leverages the monotonic relationships in the training samples. We will first review the empirical evidence supporting the monotonic relationships. Subsequently, we will detail the architecture of the PdMonoNet and then explore some theoretical aspects.

A. Empirical Analysis of Partial Correlations in $\mathcal{D}$

In $\boldsymbol{\theta}$, factors such as SNR and probability of false alarm are known to strongly correlate with detection performance, whereas the influence of other factors, such as sample rate and baud rate, is less apparent. Identifying these significant factors is a critical initial step in modeling the detection performance of practical multi-channel detection systems, a topic that has received insufficient attention in the literature. The subsequent partial correlation test demonstrates that all factors included in $\boldsymbol{\theta}$ exhibit a positive partial correlation with the label y.

Given that the feature vector $\boldsymbol{\theta} \in \mathbb{R}^{14}$ and the label $y \in \mathbb{R}$, direct computation of empirical correlations between these variables is not feasible. Therefore, we opted to perform a partial correlation test [45] for each feature in relation to the labels. This method is commonly employed in empirical research to isolate direct relationships between variables of interest, particularly when such relationships might be obscured or altered by extraneous variables.

The process involves using linear regression to mitigate the influence of confounding variables on the two variables of interest. After adjusting for these influences, the Pearson correlation coefficient is computed from the residuals.[2] The results are presented in Table I, where each reported p-value tests the null hypothesis that the corresponding partial correlation is zero.

[2]The partial correlations are calculated using `partial_corr` function in `pingouin` package [46].

The analysis reveals that all variables within the feature vector $\boldsymbol{\theta}$ demonstrate positive partial correlations with the label y in dataset $\mathcal{D}$. The extremely low p-values (less than 0.001) strongly suggest that these positive correlations are statistically significant and unlikely to have occurred by chance.

B. Architecture of PdMonoNet

We now possess empirical evidence indicating that all features in $\boldsymbol{\theta}$ exhibit positive monotonic relationships with the label y. Integrating this prior knowledge into the MLP architecture detailed in Sec. III-B presents significant challenges. The inherent structure of the MLP does not readily support the direct inclusion of monotonic information. One alternative involves the addition of regularization terms to the loss function during model training. Examples of this approach include penalizing negative gradients [47], [48] or heuristically penalizing model non-monotonicity across uniformly sampled points within the domain during training [49]. However, these regularization strategies tend to be computationally intensive and do not guarantee monotonicity across the entire domain, particularly when the test data samples fall outside the training dataset.

An alternative approach involves the development of a specialized class of neural networks designed to inherently ensure monotonicity. Examples include constrained architectures such as deep lattice networks [50] and networks with all-positive weights [51]. While these models are inherently monotonic, they often suffer from limited expressiveness or degraded performance due to their complexity. The Unconstrained Monotonic Neural Network [52] achieves guaranteed monotonicity by ensuring that the derivative function remains strictly positive, although its reliance on numerical integration for both forward and backward processes renders it impractical for high-dimensional scenarios.

Recently, the development of a Lipschitz monotonic neural network, as described in [53], offers potential for capturing the inherent monotonicity in the data. This innovative approach is elegant both theoretically and practically, offering high expressiveness without the complexity of previous methods. In the following section, we will describe the architecture of these monotonic neural networks in detail.

We present the general architecture of the proposed monotonic neural network in Fig. 2, where $\boldsymbol{\lambda} = \lambda \mathbf{1}$. This structure builds upon the framework initially described in [53], incorporating a Sigmoid activation function in the output layer to map the output to a probability between 0 and 1. The architecture is developed through the following three sequential steps:

1) Lipschitz Neural Network [54]: As detailed in Apx. B and denoted by $\tilde{g}(\boldsymbol{\theta})$, this network utilizes a Lipschitz constraint with a constant λ.
2) Basic Lipschitz Monotonic Neural Network [53]: This consists of $\tilde{g}(\boldsymbol{\theta})$ combined with a residual connection, and is denoted by
$$g(\boldsymbol{\theta}) = \tilde{g}(\boldsymbol{\theta}) + \boldsymbol{\lambda}^T \boldsymbol{\theta}. \tag{10}$$
3) Overall Monotonic Neural Network: This network includes $g(\boldsymbol{\theta})$ followed by a Sigmoid activation function,

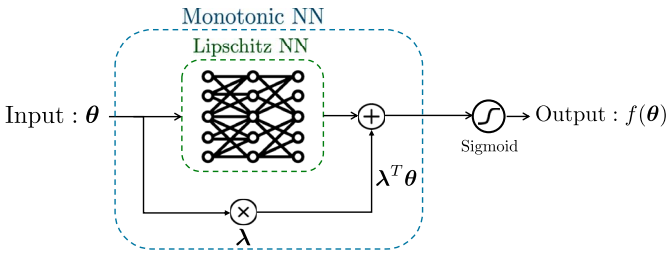


Fig. 2. Used monotonic neural network architecture for modeling the probability of detection.

which preserves the monotonic relationships, and is represented as

$$f(\boldsymbol{\theta}) = \sigma(g(\boldsymbol{\theta})). \tag{11}$$

The architecture of the Lipschitz neural network, $\tilde{g}(\boldsymbol{\theta})$, can be designed similarly to the MLP discussed in Sec. III-B. As noted in [53], $g(\boldsymbol{\theta})$ qualifies as a monotonic neural network because $\left|\frac{\partial \tilde{g}}{\partial \theta_i}\right| \leq \lambda$, ensuring that $\frac{\partial g}{\partial \theta_i} = \frac{\partial \tilde{g}}{\partial \theta_i} + \lambda$ falls within the range $[0, 2\lambda]$. This architecture is expected to be more robust and to exhibit greater generalization capabilities than the standard MLP structure, as it is designed to be monotonic and is subject to a bounded Lipschitz constraint. The monotonicity constraint acts as prior information that enhances generalization. The bounded Lipschitz constant limits the maximum rate at which the model's output can change with respect to its input, thereby reducing sensitivity to outlier data points [55].

The monotonic relationships between the parameters and detection performance may not always be strict. For instance, irrespective of the number of samples utilized, the detector might still fail due to a phenomenon known as the SNR wall [18], [56], [57]. The proposed PdMonoNet is capable to adapt to such scenarios by allowing the gradient to be zero, thus accommodating instances where increasing the number of samples does not enhance performance.

Remark 3: The monotonic relationships between θ_i and $f(\boldsymbol{\theta})$ can be adjusted flexibly by varying the λ_i values. For example, setting $\lambda_i = -\lambda$ results in $\frac{\partial g}{\partial \theta_i} = \frac{\partial \tilde{g}}{\partial \theta_i} - \lambda$, which lies in the range $[-2\lambda, 0]$, ensuring a non-increasing relationship between $f(\boldsymbol{\theta})$ and θ_i. Conversely, setting $\lambda_i = 0$ maintains $\frac{\partial g}{\partial \theta_i} = \frac{\partial \tilde{g}}{\partial \theta_i}$, spanning $[-\lambda, \lambda]$ and thus imposing no monotonicity constraints.

C. Approximation Error Analysis

Note that our used network is $f(\boldsymbol{\theta}) = \sigma(g(\boldsymbol{\theta}))$, where $g(\boldsymbol{\theta}) = \tilde{g}(\boldsymbol{\theta}) + \boldsymbol{\lambda}^T\boldsymbol{\theta}$. Then the universal approximation property of $g(\boldsymbol{\theta})$ is obviously inherited from $\tilde{g}(\boldsymbol{\theta})$, c.f. Theorem 10 in Apx. B, and summarized in the following corollary.

Corollary 4: The network $g(\boldsymbol{\theta})$, as specified in (10), is a universal approximator for any non-decreasing function processing a Lipschitz constant not exceeding 2λ.

The following Proposition 5 demonstrates that the network outlined in (11) can universally approximate the desired probability of detection, provided a mild condition is met. Specifically, this condition requires the gradient of the target function to vanish as the probability of detection approaches 0 or 1.

Proposition 5: The neural network $f(\boldsymbol{\theta})$, as specified in (11), is a universal approximator for an non-decreasing function $\ell(\boldsymbol{\theta})$ ($\ell(\boldsymbol{\theta}) \in [0, 1]$) if it satisfies

$$\left\| \frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right\|_\infty \leq 2\lambda \ell(\boldsymbol{\theta})(1 - \ell(\boldsymbol{\theta})), \forall \boldsymbol{\theta}. \tag{12}$$

Proof: Since σ is a deterministic output layer, the approximation capability of $f(\boldsymbol{\theta})$ primarily hinges on that of $g(\boldsymbol{\theta})$. The central inquiry then concerns whether $g(\boldsymbol{\theta})$ can effectively approximate $\sigma^{-1}(\ell(\boldsymbol{\theta}))$. The derivative of $\sigma^{-1}(\ell(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ is given by

$$\left\| \frac{\partial \left[\sigma^{-1}(\ell(\boldsymbol{\theta}))\right]}{\partial \boldsymbol{\theta}} \right\|_\infty = \left\| \frac{1}{\ell(\boldsymbol{\theta})(1 - \ell(\boldsymbol{\theta}))} \frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right\|_\infty \leq 2\lambda. \tag{13}$$

It demonstrates that the $\sigma^{-1}(\ell(\boldsymbol{\theta}))$ is a non-decreasing function with a Lipschitz constant no greater than 2λ. Consequently, $g(\boldsymbol{\theta})$ serves as a universal approximator for $\sigma^{-1}(\ell(\boldsymbol{\theta}))$ according to Corollary 4, which in turn establishes $f(\boldsymbol{\theta})$ as a universal approximator for $\ell(\boldsymbol{\theta})$. ■

The following Proposition 6 provides an alternative perspective on the approximation capabilities of the model without presupposing the vanishing gradient of the ground truth function. It posits that the ideal approximation error is inversely proportional to the value of the selected λ. This insight underscores the importance of the parameter λ in minimizing approximation errors, highlighting that as λ increases, the error decreases, thereby improving the model's accuracy in approximating the target function.

Proposition 6: The neural network $f(\boldsymbol{\theta})$, as specified in (11), if capable of approximating any non-decreasing function $\ell(\boldsymbol{\theta})$ ($\ell(\boldsymbol{\theta}) \in [0, 1]$) that processes a Lipschitz constant β (where $0 < \beta < \frac{\lambda}{2}$) under

$$|f(\boldsymbol{\theta}) - \ell(\boldsymbol{\theta})| \leq \frac{1}{2} - \frac{1}{2}\sqrt{1 - \frac{2\beta}{\lambda}}, \forall \boldsymbol{\theta}. \tag{14}$$

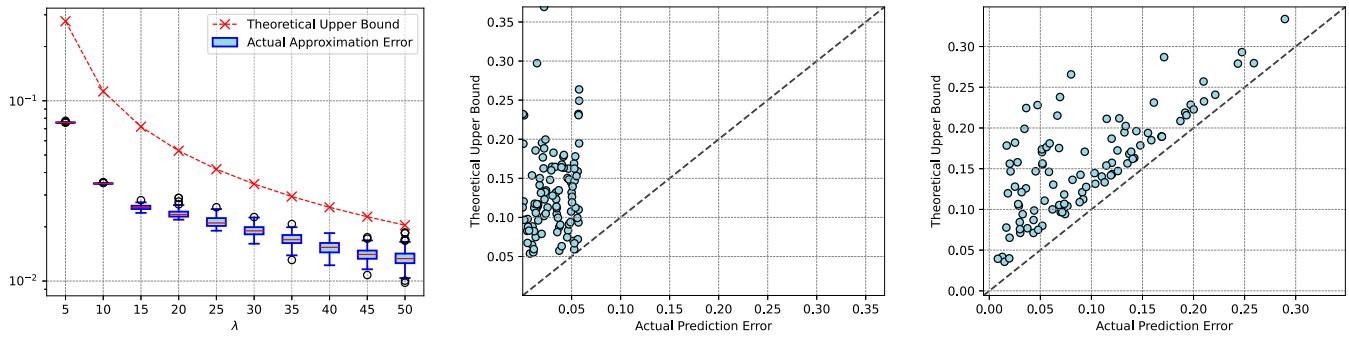
Specifically, if $\lambda \gg \beta$, then $|f(\boldsymbol{\theta}) - \ell(\boldsymbol{\theta})| \lesssim \frac{\beta}{2\lambda}$.

Proof: Define $\tilde{\ell}(\boldsymbol{\theta}) := [\ell(\boldsymbol{\theta})]_\epsilon^{1-\epsilon}$, where $\epsilon = \frac{1}{2} - \frac{1}{2}\sqrt{1 - \frac{2\beta}{\lambda}}$ is a notably small quantity (note that $\lim_{\frac{\beta}{\lambda} \to 0_+} \epsilon = \frac{\beta}{2\lambda}$). Since $\tilde{\ell}(\boldsymbol{\theta})$ retains the Lipschitz constant β from $\ell(\boldsymbol{\theta})$, we have

$$\left\| \frac{\partial \tilde{\ell}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right\|_\infty \leq \beta = 2\lambda\epsilon(1 - \epsilon) \leq 2\lambda \tilde{\ell}(\boldsymbol{\theta})(1 - \tilde{\ell}(\boldsymbol{\theta})). \tag{15}$$

According to aforementioned Corollary 4, $f(\boldsymbol{\theta})$ can universally approximate $\tilde{\ell}(\boldsymbol{\theta})$. Coupling this with the fact that $\left|\tilde{\ell}(\boldsymbol{\theta}) - \ell(\boldsymbol{\theta})\right| \leq \epsilon$, we conclude $|f(\boldsymbol{\theta}) - \ell(\boldsymbol{\theta})| \leq \epsilon$. ■

We validate Proposition 6 using a toy example. A dataset of 1,000 samples is generated, where the feature x is uniformly sampled from $[-2, 2]$, and the label is computed as $\ell(x) = 2x + 0.5$, truncated to $[0, 1]$. The target function has a Lipschitz constant of $\beta = 2$. The network is trained for 10,000 epochs, with the process repeated 100 times for each λ. The final approximation errors are compared to the theoretical bounds in



(a) Validation of Proposition 6. (b) Validation of Proposition 7 ($\lambda = 10$). (c) Validation of Proposition 8 ($\lambda = 10$).

Fig. 3. Numerical validation of the theoretical analysis. The PdMonoNet model is configured with an input dimension of 1 and hidden layer sizes [8, 16, 8, 16].

Proposition 6. As shown in Fig. 3(a), the error decreases as λ increases and remains below the theoretical upper bound. However, larger λ empirically increases training costs, requiring more epochs for convergence. To address this, sufficient training epochs must be ensured when using a large λ. Alternatively, λ can be increased incrementally, with the model trained for a limited number of epochs at each step until the training loss stabilizes.

D. Prediction Error Analysis

Consider the network $\hat{f}$, a well-trained PdMonoNet using the dataset $\mathcal{D}$ that utilizes the architecture described in (11). Our objective is to quantify the prediction error when this network is applied to an unseen data sample. We postulate that the relationship between features and labels in $\mathcal{D}$ can be modeled as $y_i = \ell(\boldsymbol{\theta}_i) + n_i$, where ℓ represents the true underlying relationship the neural network aims to learn, and n_i is the independent measurement noise. Leveraging the beneficial properties of Lipschitz functions, we can establish a bounded prediction error for this scenario.

Proposition 7: Given that ℓ is function with a Lipschitz constant β, the prediction error of $\hat{f}$ at an unseen feature point $\boldsymbol{\theta}_t$ is bounded by

$$\left|\hat{f}(\boldsymbol{\theta}_t) - \ell(\boldsymbol{\theta}_t)\right| \leq \min_{i=1,\ldots,|\mathcal{D}|} \left\{(2\lambda+\beta)\left\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_i\right\|_1 + \underbrace{\left|\ell(\boldsymbol{\theta}_i) - y_i\right|}_{\text{measurement error}} + \underbrace{\left|\hat{f}(\boldsymbol{\theta}_i) - y_i\right|}_{\text{training error}}\right\} \quad (16)$$

Proof: See Apx. C. ∎

Proposition 7 is also validated using the toy example. The PdMonoNet model is trained on the same dataset used in Fig. 3(a), but with added Gaussian noise. A test dataset is then generated following the same procedure. The trained PdMonoNet is applied to this test dataset, and the prediction errors on the unseen data are recorded and compared to the theoretical bounds established in Proposition 7. As shown in Fig. 3(b), the observed prediction errors consistently fall below the theoretical bounds.

The Proposition 7 delineates that the prediction error bounds can be decomposed into three components: the distance between the unseen feature and any feature in $\mathcal{D}$, the measurement error arising during the data sampling phase, and the training error incurred during the training phase. Technically, while the fitting error is quantifiable, the measurement error remains unknown but can be mitigated by increasing the number of Monte Carlo simulations, as discussed in Fact 1 and Fact 2.

Leveraging the monotonic nature of PdMonoNet, we can establish an alternative form of prediction error bound, as described in Proposition 8.

Proposition 8: Suppose that ℓ is a non-decreasing function. For an unseen feature point $\boldsymbol{\theta}_t$, if there exist $\boldsymbol{\theta}_l, \boldsymbol{\theta}_u \in \mathcal{D}$ such that $\boldsymbol{\theta}_l \leq \boldsymbol{\theta}_t \leq \boldsymbol{\theta}_u$, the prediction error of $\hat{f}$ at $\boldsymbol{\theta}_t$ is bounded by

$$\begin{aligned}&\left|\hat{f}(\boldsymbol{\theta}_t) - \ell(\boldsymbol{\theta}_t)\right| \\ &\leq |y_l - y_u| + \max\Big\{\underbrace{\left|\ell(\boldsymbol{\theta}_u) - y_u\right|}_{\text{measurement error}} + \underbrace{\left|\hat{f}(\boldsymbol{\theta}_l) - y_l\right|}_{\text{training error}}, \\ &\qquad \underbrace{\left|\ell(\boldsymbol{\theta}_l) - y_l\right|}_{\text{measurement error}} + \underbrace{\left|\hat{f}(\boldsymbol{\theta}_u) - y_u\right|}_{\text{training error}}\Big\} \end{aligned} \quad (17)$$

Proof: See Apx. D. ∎

Proposition 8 is also validated using the toy example. The PdMonoNet model is trained on the dataset generated similarly to the one in Fig. 3(a), with added Gaussian noise, but excluding data samples with features in the range $[-0.01, 0.01]$. This process is repeated 100 times, each with a newly generated training dataset. The maximum prediction error within the excluded interval $[-0.01, 0.01]$ is recorded and compared to the theoretical bounds provided in Proposition 8. As shown in Fig. 3(c), the observed prediction errors consistently remain below the theoretical bounds.

This proposition illustrates that, with controllable measurement error and known training error, the prediction error for any unseen point $\boldsymbol{\theta}_t$ within the interval $[\boldsymbol{\theta}_l, \boldsymbol{\theta}_u]$ can be bounded by the term $|y_l - y_m|$. This characteristic is particularly beneficial as it permits sparse sampling when the outputs change smoothly over certain ranges.

Another critical aspect is the quantification of in-sample prediction error. During the training process, the following inequality holds for any $(\boldsymbol{\theta}_i, y_i) \in \mathcal{D}$, as derived using the triangle

inequality:

$$\underbrace{\left|\hat{f}(\boldsymbol{\theta}_i)-\ell(\boldsymbol{\theta}_i)\right|}_{\text{in-sample prediction error}} \geq \underbrace{\left|\hat{f}(\boldsymbol{\theta}_i)-y_i\right|}_{\text{training error}} - \underbrace{\left|\ell(\boldsymbol{\theta}_i)-y_i\right|}_{\text{measurement error}}. \quad (18)$$

The training error is directly observable during the training process. To estimate the measurement error, which depends on the distribution of the dataset, we provide its numerical approximation through the following Proposition 9.

Proposition 9: Assuming that the true label distribution in dataset $\mathcal{D}$ adheres to the probability density function $p(y)$, the average measurement error of $\mathcal{D}$, denoted by $m(\mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{\theta}_i,y_i)\in\mathcal{D}}|\ell(\boldsymbol{\theta}_i)-y_i|$, can be approximated by:

$$m(\mathcal{D}) \approx \int_0^1 \sqrt{\frac{2}{\pi}}\sqrt{\frac{x(1-x)}{M}}p(y)dy. \quad (19)$$

Proof: This is straightforwardly derived from Fact 2. ■

Given that the components of the inequality (18) can be numerically determined during the training process, they provide a real-time estimate of the lower bound for the in-sample prediction error. This estimate may serve as a useful indicator for determining an appropriate stopping point for training, such as when the calculated error bound approaches or falls below zero.

E. Computational Complexity Analysis

In this subsection, we provide a detailed analysis of the computational complexity of our proposed PdMonoNet and compare it with the traditional Monte Carlo simulation approach.

The computational cost of PdMonoNet is determined by its architecture, as described in Section IV-B. Assuming the input dimension is I and there are D hidden layers, with h_i neurons in the i-th layer, the computational complexity of an inference process through the entire network is given by $\mathcal{O}(I + Ih_1 + \sum_{i=1}^{D-1} h_i h_{i+1} + h_D)$. In comparison, the Monte Carlo simulation approach, detailed in Section III-A, primarily incurs computational costs from evaluating the test statistics, $T(\mathbf{X})$, as indicated in Equations (6) and (7), repeated $K+M$ times. Assuming the computational cost for a single evaluation of $T(\mathbf{X})$ is $\mathcal{O}(S)$, the total computational cost of a single evaluation of the probability of detection is $\mathcal{O}((K+M)S)$.

It is important to note that even using a relatively simple method such as energy detection, where $T(\mathbf{X}) = \|\mathbf{X}\|_F^2$, the computational cost is already $\mathcal{O}(NL)$. For more complex methods like the GLRT approach, the cost significantly exceeds NL due to iterative matrix eigenvalue decomposition operations [24].

The computational burden of the Monte Carlo method is particularly notable, as the parameters K, M, and L typically take on large values to produce reliable results. In contrast, once trained, our proposed PdMonoNet can efficiently generate results. Its computational complexity increases linearly with the number of layers (D) and feature dimensions (I), which are generally much smaller than K or M. This efficiency offers a significant advantage of our PdMonoNet over traditional methods in terms of computational expense.

TABLE II
THE EMPIRICAL PARTIAL CORRELATION IN DATASET $\mathcal{T}$

Feature	Notation	Coeff.	95% CI	p-value
Sample Rate	f_s	-0.88	[-0.88, -0.88]	< 0.001
Prob. of False Alarm	P_{FA}	-0.14	[-0.15, -0.14]	< 0.001
Observation Time	t	-0.78	[-0.79, -0.78]	< 0.001
Number of Receivers	L	0.61	[0.61, 0.62]	< 0.001

V. APPLYING MONOTONIC NEURAL NETWORK TO THRESHOLD DETERMINATION

In this section, we explore another application of the monotonic neural network: estimating the threshold γ. Although the threshold primarily indicates the probability of a false alarm and does not directly affect the overall detection performance, it remains essential for detecting the presence of the primary signal in each task. The theoretical calculation of this threshold, as shown in (3), presents significant challenges, as discussed in Sec. II-B. Typically, the threshold is estimated using Monte Carlo simulation (see (6)), a method that is computationally expensive. Therefore, mirroring our approach to predicting the probability of detection, we opt to estimate the threshold using a monotonic neural network.

A. Dataset Generation and Processing

The required dataset for threshold estimation can be collected during the data generation process described in Apx. A. Since the features related to primary signal generation do not influence the threshold estimation (noting that threshold estimation occurs when the primary signal is absent), we collect features such as the sampling rate of the receivers, the probability of false alarm, the observation time, and the number of receivers. Because thresholds are typically very small positive numbers, we use their log-scaled values as labels. This results in a dataset specifically for threshold estimation, denoted as $\mathcal{T}$.

B. Architecture Design

We also perform an empirical analysis of the partial correlations between features and labels in the dataset $\mathcal{T}$. As detailed in Table II, with the exception of the number of receivers, all other features exhibit negative partial correlations with the thresholds. Consequently, as discussed in Remark 3, we adapt the architecture of the monotonic neural network shown in Fig. 2. We remove the last Sigmoid output layer and set $\boldsymbol{\lambda} = [-\lambda, -\lambda, -\lambda, \lambda]^T$. For convenience, we have named the MNN-based approach for estimating thresholds as ThreshMonoNet, and the MLP-based approach as ThreshNet.

VI. NUMERICAL EXPERIMENTS

In this section, we conduct experiments to evaluate the performance prediction and threshold determination for multichannel detection problems. Firstly, we demonstrate the predictive capabilities of our proposed PdMonoNet on both multichannel energy detection and multichannel GLRT detection. Subsequently, we illustrate the threshold determination capabilities

TABLE III
COMPARISON OF DIFFERENT METHODS WITH THEIR CAPABILITIES

Method	Monotonicity	Advantages	Disadvantages	Most Applicable Tasks
APMNN	Full	Guarantees global monotonicity	Limited flexibility and approximation ability	Simple scenarios requiring strict monotonic relationships
RBF	None	Efficient in low-dimensional spaces	Poor scalability to high dimensions; sensitive to the choice of kernel parameters	Function approximation, regression, and classification in low-dimensional spaces
ELM	None	Extremely fast training due to closed-form solution	No monotonicity guarantee; limited generalization for complex problems	Scenarios requiring fast training and inference
PdNet	None	High efficiency	No monotonicity guarantee	Scenarios without monotonic relationships
PdMonoNet	Flexible	Supports partial monotonicity	May underperform if λ is too small	Complex scenarios with mixed monotonic and non-monotonic relationships

of our proposed ThreshMonoNet for multichannel GLRT detection. Finally, we validate our findings through the application of multichannel spectrum sensing on Software-Defined Radio (SDR) devices.

To ensure a fair comparison between our proposed models and the benchmarks, we minimize the influence of hidden layer sizes and the learning process. Suitable hidden layer sizes are identified through preliminary experimental trials, and the neural networks are trained using the Adam optimizer [58] with reduced learning rates.

A. Comparison on Performance Prediction

In this part, we conduct the performance comparison between our proposed PdNet and PdMonoNet models, focusing on their ability to predict the probability of detection. The evaluation involves two widely-used multichannel detection methods: the energy detector and the GLRT detector. For benchmarking purposes, we include the all-positive weight monotonic neural network (APMNN) [51], the radial basis function (RBF) neural network [59], and the extreme learning machine (ELM) [60] in our comparison. Table III summarizes the advantages and disadvantages of each method, providing a comprehensive comparison.

1) Multichannel Energy Detection: The detection performance of multichannel energy detection can be described using mathematical expressions. For instance, when an equal gain combination scheme is applied, the corresponding detection performance expression for multichannel energy detection is given as [56, Equation (8)]:

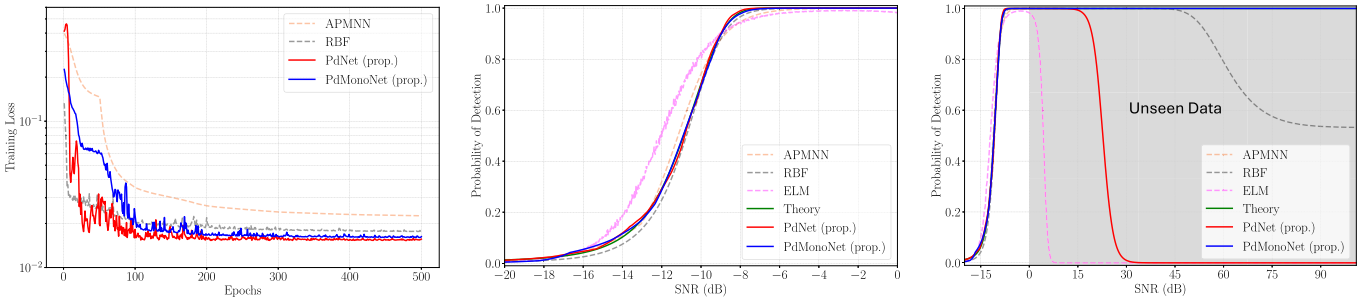
$$P_D = Q\left(\frac{Q^{-1}(P_{FA}) - \sqrt{\frac{N}{2L}}\sum_{l=1}^{L}\gamma_l}{\sqrt{\frac{1}{L}\sum_{l=1}^{L}(1+2\gamma_l)}}\right), \tag{20}$$

where $Q(x) = \int_x^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\, dt$ and $\gamma_l > 0$ represents the SNR at the l-th receiver.

To generate the training dataset, we create 10,000 samples featuring P_{FA} and γ_l for two receivers. Here, P_{FA} is uniformly selected from 0.1% to 2.0%, and γ_1 and γ_2 range independently between -20 dB and 0 dB. The labels are initially computed theoretically using (20) and then refined through Monte Carlo sampling over 100 experiments to replicate the dataset generation process for $\mathcal{D}$.

In this part, the PdNet model is configured with hidden layers arranged as [24, 48, 24, 48]. The PdMonoNet model shares the same hidden layer sizes as the PdNet, with λ set to a sufficiently large value of 100. The APMNN model configuration consists of 128 groups, each containing 8 neurons. Each RBF layer in the RBF model employs a Gaussian basis function to transform inputs based on radial distance from the center, with 256 centers in the hidden layer. The ELM model is configured with 256 neurons in its hidden layer. We configured the input dimension to three to align with the task requirements and applied the sigmoid function to the final outputs of all models to facilitate the prediction of detection probabilities. Parameters for the ELM model are directly derived from the solution of the least squares problem, whereas the parameters for other models are acquired through stochastic optimization using the Adam optimizer [58]. The loss function is chosen to be the ℓ_1 norm. The learning rate is reduced by a factor of 0.5 every 100 epochs. The optimal hyperparameters are determined through a grid search that varied batch sizes (2^8, 2^9, 2^{10}, 2^{11}) and learning rates (0.02, 0.01, 0.005, 0.001).

In Fig. 4, we illustrate the training loss and learned probability of detection across various methods under different SNR conditions. Fig. 4(a) reveals that all methods demonstrate a decreasing trend in training loss with increasing epochs. Notably, our proposed PdNet achieves the lowest training loss, closely followed by PdMonoNet, which despite sharing PdNet's hidden layer sizes, incurs a slightly higher training loss due to its monotonic constraint. This constraint restricts PdMonoNet from fitting data that violate the monotonicity due to measurement errors, whereas PdNet can adjust more flexibly to such errors. The RBF and APMNN models also show competent training losses, albeit higher than those of PdNet and PdMonoNet, indicating effective in-sample data approximation. Fig. 4(b) depicts the predicted probability of detection versus SNR changes within the -20dB to 0dB range used for training. The PdNet and PdMonoNet models exhibit the best in-sample approximation, aligning with their training loss performance. Although the APMNN and RBF models perform slightly less effectively, the simpler ELM method trails behind, likely due to its inability to capture complex data patterns and dependencies.

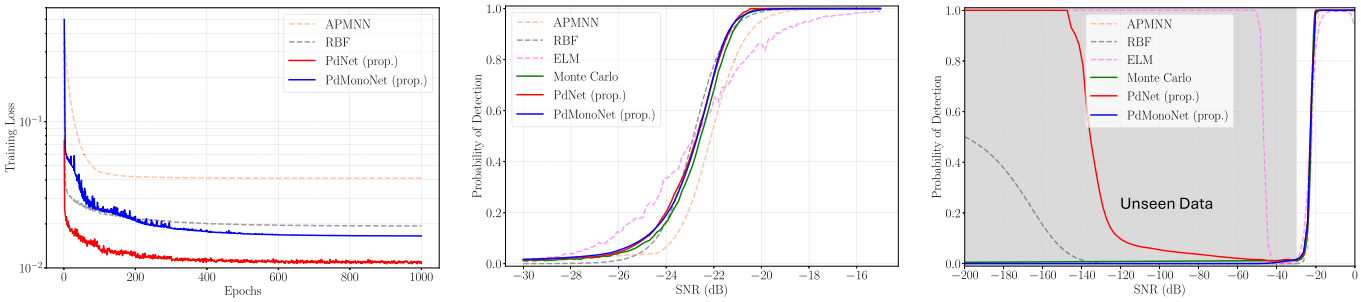


(a) Variation of training loss across epochs.

(b) Comparison of model performance within the training region.

(c) Comparison of model performance outside the training region.

Fig. 4. Training loss and learned probability of detection by different methods under varying SNR conditions for two receivers with $P_{FA} = 1\%$.



(a) Variation of training loss across epochs.

(b) Comparison of model performance within the training region.

(c) Comparison of Model Performance Outside the Training Region.

Fig. 5. Training loss and learned probability of detection by different methods under varying SNR conditions, with parameters: $f_b = 10$ kHz, $f_s = 40$ kHz, $t = 0.5$ s, $P_{FA} = 1\%$, and 6 receiving nodes.

Fig. 4(c) explores predictions over an extended SNR range of -20dB to 100dB, beyond the training dataset. Here, the non-monotonic models, i.e., RBF, ELM, and PdNet, show an abnormal decrease in detection probability. In contrast, the APMNN and PdMonoNet models maintain reliable outputs, benefiting from the monotonic constraints that enhance their performance under extrapolated conditions. Furthermore, it is worth noting that the prediction error between the theoretical results and the outputs from PdMonoNet is bounded by our proposed Proposition 7 and 8. This constitutes another advantage of our proposed PdMonoNet.

2) Multichannel GLRT Detection: The detection performance of the GLRT method is considerably more complex than that of the energy detection. As previously discussed, we have prepared a dataset, denoted as $\mathcal{D}$, as described in Apx. A. This dataset records the feature settings as environmental parameters and the labels as the probability of detection for the GLRT method. We have also included the results from Monte Carlo simulations, which are considered as the ground truth in our analysis. These simulations are conducted with a substantial number of realizations: $K = 10^5$ and $M = 10^4$. This significant sample size tries to ensure the statistical accuracy and reliability of the simulation outcomes.

Since the input features for this task have a higher dimensionality compared to the task in Section VI-A1, we appropriately increase the hidden layer sizes for each model. For the PdNet and PdMonoNet models, the hidden layers are configured as $[112, 224, 112, 224, 112, 224]$. The hyperparameter λ for PdMonoNet is set to a large value of 100. The APMNN model configuration includes 512 groups, each consisting of 16 neurons. Each layer in the RBF model uses a Gaussian basis function to transform inputs according to the radial distance from the center, with 1024 centers in its hidden layer. The ELM model incorporates 1024 neurons in its hidden layer. We set the input dimension to 14 to meet the requirements of the task and applied a sigmoid function to the final outputs of all models to facilitate the prediction of detection probabilities. Parameters for these models, with the exception of the ELM model, are optimized using the Adam optimizer, with optimal hyperparameters determined through a grid search that varied batch sizes $(2^{10}, 2^{11}, 2^{12}, 2^{13})$ and learning rates (0.02, 0.01, 0.005, 0.001). The loss function is chosen to be the ℓ_1 norm. The learning rate is reduced by a factor of 0.5 every 100 epochs.

In Fig. 5, we illustrate the training loss and learned probability of detection for different methods under different SNR conditions. Fig. 5(a) demonstrates a decreasing trend in training loss with increasing epochs across all methods. The performance hierarchy is similar to that observed in Fig. 5(a), where our proposed PdNet and PdMonoNet outperform others. Fig. 5(b) shows that all models perform reasonably well under the parameter variations included in the training set, with our

TABLE IV
COMPUTATIONAL TIME FOR DIFFERENT METHODS

	APMNN	RBF	ELM	PdNet (prop.)	PdMonoNet ($\lambda = 100$) (prop.)
Total Training Time (Seconds, 1000 Epochs)	9742	10943	21	3460	4204
Training Time per Epoch (Seconds)	9.74	10.94	–	3.46	4.20
Inference Time per Sample (Milliseconds)	4.91	1.51	0.20	0.81	0.83

proposed PdNet and PdMonoNet exhibiting the best approximation performance. The performance disparity between the proposed PdMonoNet and the Monte Carlo simulation results can be attributed to several factors. These include the approximation error of PdMonoNet, which is influenced by its architecture, the hyperparameter λ, and the quality of training dataset, as well as the inherent randomness of the Monte Carlo simulations. The APMNN exhibits a significant performance gap compared to Monte Carlo methods, likely due to its limited capacity to model complex functions. It has been commonly recognized that imposing overly restrictive constraints, such as requiring all-positive weights, significantly limits the hypothesis space for weight parameters [53]. Consequently, APMNN often performs poorly when modeling complex functions. In contrast, PdMonoNet avoids such limitations by employing a more flexible yet principled design, enabling it to better capture complex functional relationships while maintaining interpretability. The ELM remains the least effective among all the methods. However, as seen in Fig. 5(c), the RBF, ELM, and PdNet exhibit non-monotonic behavior. In contrast, PdMonoNet consistently delivers more accurate and logical results. This reliability is crucial not only for accuracy but also for solving optimization problems that utilize network outputs as objective functions. For example, using PdNet outputs as objectives in first-order optimization methods could lead the system to erroneously pursue lower SNR settings, an issue avoided by employing the more logical PdMonoNet outputs.

In Table IV, we compare the computational time of these methods during both the training and inference stages. The results show that the ELM method is the fastest overall, while our proposed PdNet and PdMonoNet methods are more efficient than the APMNN and RBF methods. Notably, increasing λ in PdMonoNet does not result in a more complex network structure and, therefore, does not increase the computational time per epoch during training or inference. However, a larger λ may require more training epochs for the loss to stabilize. Thus, sufficient training epochs must be ensured when using a large λ.

B. Comparison on Threshold Determination

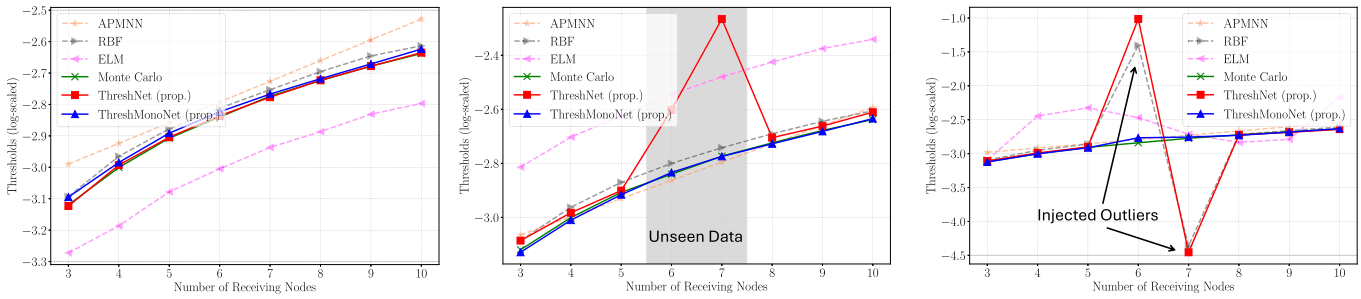
In this part , we present a detailed comparison of our proposed ThreshNet and ThreshMonoNet models in threshold determination, highlighting their robustness in handling missing data and outliers in the training dataset.

The ThreshNet model employs hidden layers configured as $[50, 100, 50, 100, 50, 100]$. ThreshMonoNet shares these dimensions with ThreshNet and maintains a high hyperparameter, λ, set at 10. The APMNN features 128 groups, each with 8 neurons, while each RBF layer utilizes Gaussian basis functions with 256 centers to transform inputs radially. The ELM model includes 256 neurons in its hidden layer. We configured the input dimension to 4 to align with the task requirements. Parameters for all models, except the ELM, are optimized using the Adam optimizer, with optimal hyperparameters determined through grid search involving batch sizes (2^8, 2^9, 2^{10}, 2^{11}) and learning rates (0.02, 0.01, 0.005, 0.001).

Fig. 6(a) illustrates how the learned thresholds vary with the number of receiving nodes, showcasing results from several models. Notably, all methods, except the ELM method, effectively approximate the thresholds under the parameter variations included in the training set, with our ThreshNet and ThreshMonoNet demonstrating superior performance.

The robustness of our models is further assessed in Fig. 6(b), where data for 6 and 7 node configurations were intentionally excluded from the training set. Under these conditions, ThreshNet shows significant estimation errors, indicating a lack of generalization when data is incomplete. In contrast, APMNN, RBF, and ThreshMonoNet maintain accuracy, with ThreshMonoNet performing the best. The stable performance of the RBF model is likely due to its localized Gaussian basis functions, which enable effective interpolation and robust generalization, even with missing data. The monotonicity embedded in ThreshMonoNet and APMNN likely aids in maintaining consistent threshold predictions, even with missing data, suggesting their suitability for applications with sparse data collection.

In scenarios with significant outliers, as depicted in Fig. 6(c), the distinction between non-monotonic neural networks and monotonic neural networks becomes more evident. Outliers were introduced by setting artificially extreme threshold values for certain node configurations. The performance of the RBF, ELM, and ThreshNet is negatively impacted as they attempt to accommodate these outliers, leading to a model that fits these inaccurate points at the expense of overall accuracy. In contrast, the performance of the APMNN and ThreshMonoNet remains stable and closely aligns with the Monte Carlo simulation results, emphasizing their robustness to extreme data points. The imposed monotonic constraint prevents these models from being influenced by outliers, ensuring that they maintain a logical order in their predictions. Furthermore, it is noteworthy that our proposed ThreshNet approximates the desired results with the highest accuracy.



(a) Case I: Both models performs well.

(b) Case II: The ThreshNet exhibits significant errors when processing inputs that it has never encountered before.

(c) Case III: The ThreshNet tries to learn the outliers while the ThreshMonoNet does not.

Fig. 6. Learned thresholds by different methods under varying number of receiving nodes, with parameters: $f_s = 20$ kHz, $t = 0.5$ s, $P_{FA} = 1\%$, and (a) neural networks are trained with complete and accurate dataset; (b) neural networks are trained with an incomplete dataset, missing data for 6 and 7 node configurations (c) neural networks are trained with a contaminated dataset where thresholds were artificially set to 10^{-1} for the 6 node configuration and $10^{-4.5}$ for the 7 node configuration.

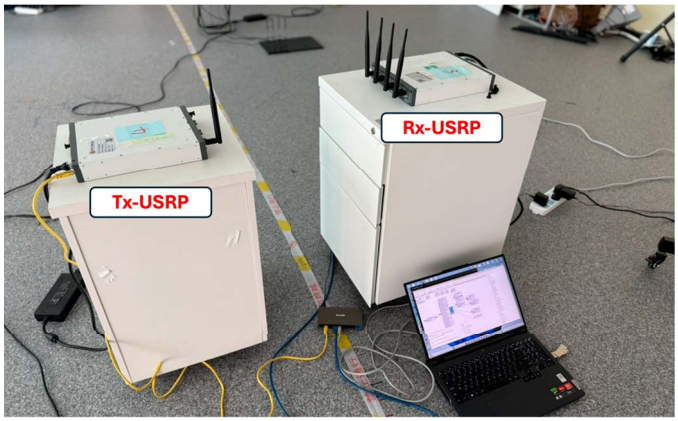


Fig. 7. Experiment platform setup.

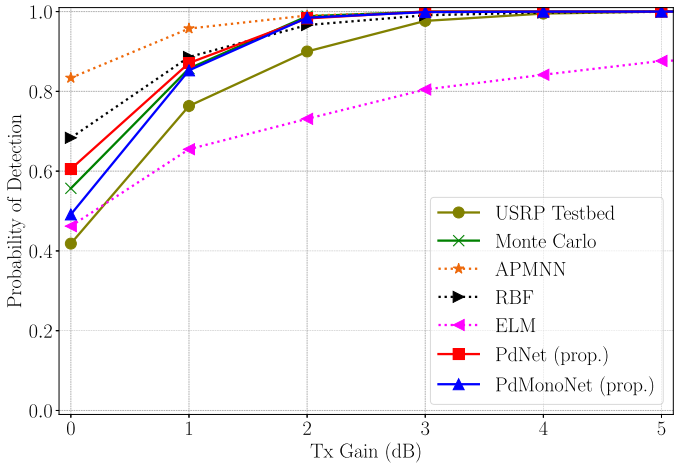


Fig. 8. Prediction of detection performance on software-defined radio testbed by different methods.

C. Real Data Experiments on SDR Testbed

In this part, we validate our proposed methods for predicting detection performance of the multichannel spectrum sensing application using real data collected from a software-defined radio (SDR) testbed. Our laboratory experiment utilizes two NI Universal Software Radio Peripherals (USRP) X410 models: one equipped with a single antenna serving as the transmitter and the other equipped with four antennas acting as a multi-antenna receiver. To simulate a weak signal scenario, the transmitter is connected to a 30dB attenuator.

Fig. 7 illustrates the experimental setup, which adheres to the parameters used to generate dataset $\mathcal{D}$ as described in Apx. A. Specifically, the transmitter emits signals at a carrier frequency of 1.4GHz using randomly-generated Quadrature Phase Shift Keying (QPSK) signals at a baud rate of $f_b = 10$ kHz, with the transmitter gain set to 0 dB. Operating at a sampling rate of $f_s = 20$ kHz and a receiving gain of 50 dB. The data collected is then transmitted to a personal computer via a wired connection. Each detection experiment captures consecutive samples over a $t = 0.2$ second duration. The detection threshold for each method is calculated using 1000 realizations of recorded noise data to achieve a false alarm rate $P_{FA} = 1\%$.

Fig. 8 shows the detection probabilities of different methods (obtained in Sec. VI-A2) as a function of transmitter gain. Additionally, Table V presents the mean absolute error (MAE) of each method when compared to the Monte Carlo results and the USRP test results. The required input SNR data for these models were calibrated under a condition of 30dB transmitter gain and adjusted based on the actual transmitter gain. A slight discrepancy exists between the detection performances from Monte Carlo simulations using synthetic data and those derived from the real data experiment. This discrepancy is likely caused by the multipath propagation characteristics of the indoor environment, which differ from the conditions used to synthesize the training dataset $\mathcal{D}$, where an ideal single-path scenario is assumed.

Our proposed models, PdNet and PdMonoNet, demonstrate the closest alignment with the Monte Carlo results. Notably, all methods provide a reasonable approximation of the Monte Carlo outcomes, validating their effectiveness in predicting performance for this multi-channel spectrum sensing system. Among these, PdMonoNet achieves the most accurate prediction results. However, non-monotonic networks, such as RBF,

TABLE V
MEAN ABSOLUTE ERRORS FOR DIFFERENT METHODS COMPARED TO MONTE CARLO RESULTS AND USRP TEST RESULTS

	APMNN	RBF	ELM	PdNet (prop.)	PdMonoNet (prop.)
Monte Carlo	0.029	0.015	0.137	**0.005**	**0.005**
USRP Testbed	0.056	0.036	0.116	0.031	**0.021**

ELM, and PdNet, are anticipated to become unreliable under extreme SNR conditions, further emphasizing the superior robustness and performance of the proposed PdMonoNet.

VII. CONCLUSION AND FUTURE DIRECTIONS

In conclusion, this paper makes significant contributions to the field of multichannel detection by critically evaluating the prevalent use of Monte Carlo simulations to assess detection performance and thresholds. We have effectively deployed a basic multilayer perceptron (MLP)-based neural network, referred to as PdNet, as an initial method for efficiently predicting detection performance. Additionally, we introduced a novel measurement technique using a monotonic neural network (MNN), named PdMonoNet, which has shown superior performance compared to PdNet due to its ability to leverage the inherent monotonic relationships within the dataset. These approaches were further developed into MLP-based ThreshNet and MNN-based ThreshMonoNet for threshold determination. Moreover, we explored the theoretical underpinnings of our PdMonoNet approach, focusing on its universal approximation capabilities and its predictive accuracy with unseen data. Our comprehensive numerical experiments have underscored the effectiveness of our proposed methods, demonstrating their guaranteed monotonicity and resilience against outliers and new data points.

Our findings highlight the potential of using monotonic neural networks for pre-estimating system performance, which significantly benefits techniques such as dynamic control of complex systems. This opens up avenues for further research into predicting the performance of distributed localization and tracking systems. Another promising direction for future research is to explore the properties of monotonic neural networks, such as deriving their sample complexity, which is crucial for practical applications.

APPENDIX

A. Generation Process of Dataset $\mathcal{D}$

1) The Overall Structure of the Dataset: The dataset features, denoted as $\boldsymbol{\theta}$, are carefully selected to capture critical aspects of the signal environment and receiver settings. Drawing on a review of related literature [24], [30], [61], typical features include the baud rate of target signals, the sampling rate of receivers, the probability of false alarm, observation time, and the signal-to-noise ratio (SNR) for each of up to ten receivers. These features are instrumental in quantifying signal quality across various noise conditions and channel models. Non-existent receivers are represented by setting their SNR values to -100 dB.

Data labeling involves generating output variables, y, representing the probability of detection by the GLRT detector, with values ranging from 0 to 1. These probabilities are derived from empirical Monte Carlo simulations, as detailed in Sec. III-A. It is important to note that y is derived from Monte Carlo experiments; therefore, it corresponds to $\ell(\boldsymbol{\theta})$ plus unknown measurement noise. The feature also includes the probability of false alarm, which is used to empirically estimate threshold values $\hat{\gamma}$ as shown in (6).

The resultant dataset, $\mathcal{D} = \{(\boldsymbol{\theta}_i, y_i)\}_{i=1}^{N}$, contains over $N = 10^5$ data samples, compiled according to the procedure outlined in the next subsection. Due to the substantial computational burden,[3] we have to set $K = 1000$ and $M = 100$ for generating $\mathcal{D}$. While the setting also encompasses categorical data such as signal modulation type and detection algorithms, this paper excludes such data as they are not numerical and lack a meaningful order or magnitude, which is necessary for MLPs. Although categorical data can be numerically encoded, this often oversimplifies the data, potentially losing valuable information. Therefore, this study focuses solely on inherently numerical features.

2) The Signal Synthetic Process: In the signal generation phase, an environment encompassing p distributed CR users. Each of these users is furnished with an individual omnidirectional antenna. The primary objective here is to identify a sole primary source with a single antenna, specified as $r = 1$. The primary signal in question is identified as transmitting Quadrature Phase Shift Keying (QPSK) modulated signals, with the baud rate set precisely to f_b Hz. Each receiver operates at a sampling rate of f_s Hz. The channel between each receiver and the signal source is hypothesized to be an independent Rician fading channel with K-factor being 4. These modulated signals are generated utilizing the MATLAB Communications Toolbox, as referenced in [62]. The signal-to-noise ratio (SNR) for the received signal are independent for each antenna. Each simulation experiment is executed during t consecutive duration. The threshold γ is calculated to match the probability of false alarm P_{FA} using the empirical test statistics from 1000 realizations of pure noise data. The probability of detection is determined by repeating the detection procedure 100 times.

For each sample in $\mathcal{D}$, parameters are uniformly selected as follows: f_b ranges from 1 kHz to 20 kHz, f_s from $2f_b$ to $10f_b$, t from 0.1 s to 1 s, P_{FA} from 0.1% to 2.0%, and the number of receiving nodes from 3 to 10. The SNR for each node is independently set between -40 dB and 0 dB.

B. A Lipschitz Neural Network with Universal Approximation Property

As introduced in [63], [64], a fully connected scalar-valued network, denoted as $f(\mathbf{x})$, is considered Lipschitz bounded by a constant λ if it fulfills the condition $\|\nabla f(\mathbf{x})\|_\infty \leq \lambda$, $\forall \mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^I$ is the input vector. An effective method to construct a

[3]The generation of $\mathcal{D}$ required more than two weeks on our server, which is equipped with 96 CPU cores.

Lipschitz bounded neural network is by restricting the matrix norm of all weight matrices. This method is detailed below.

Note that the $f(\mathbf{x})$ of depth D can be defined as

$$\mathbf{x}^d = \mathbf{W}^d\sigma(\mathbf{x}^{d-1}) + \mathbf{b}^d, \quad d = 1, \ldots, D \tag{21}$$

where $\mathbf{x}^d$ denotes the output from the d-th layer ($\mathbf{x}^0$ is the input and $\mathbf{x}^D$ is the output), $\mathbf{W}^d$ represents the weight matrix, $\mathbf{b}^d$ is the bias vector, and σ is a non-linear activation function applied element-wise and has a Lipschitz constant no greater than 1. Then the ℓ_∞ norm of gradient of $f(\mathbf{x})$ on $\mathbf{x}$ admits the following bounds

$$\|\nabla f(\mathbf{x})\|_\infty \leq \left\|\Pi_{d=1}^D \mathbf{W}^d\right\|_\infty. \tag{22}$$

Thus, simply imposing $\left\|\Pi_{d=1}^D \mathbf{W}^d\right\|_\infty \leq \lambda$ will also lead to $\|\nabla f(\mathbf{x})\|_\infty \leq \lambda$, which ensure the λ as the Lipschitz constant of neural network (21). One of the options to keep $\left\|\Pi_{d=1}^D \mathbf{W}^d\right\|_\infty \leq \lambda$ is to impose $\left\|\mathbf{W}^1\right\|_{1,\infty} \cdot \Pi_{d=2}^D \left\|\mathbf{W}^d\right\|_\infty \leq \lambda$ [53], [54], which can be conveniently achieved by simple normalization of each weight matrix as detailed in [53]. It is essential to recognize that λ is a constant hyperparameter that must be predefined by the user.

The following Theorem 10 shows the universal approximation property of this type of Lipschitz neural network to any Lipschitz continuous function.

Theorem 10 [54, Theorem 3] (Universal Approximation with Lipschitz Networks): For a fully-connected networks described by (21), if σ is GroupSort activation having a group size of 2 and its weights matrices are constrained as $\left\|\mathbf{W}^1\right\|_{1,\infty} \cdot \Pi_{d=2}^D \left\|\mathbf{W}^d\right\|_\infty \leq \lambda$, then this network can approximate any Lipschitz function with its Lipschitz constant being λ.

C. Proof of Proposition 7

Note that the used neural network f is designed with maximum Lipschitz constant 2λ. Given any data sample $(\boldsymbol{\theta}_i, y_i) \in \mathcal{D}$, the prediction error of $\hat{f}$ at an unseen feature point $\boldsymbol{\theta}_t$ is bounded as follows

$$\begin{aligned}
&\left|\hat{f}(\boldsymbol{\theta}_t) - \ell(\boldsymbol{\theta}_t)\right| \\
&= |\hat{f}(\boldsymbol{\theta}_t) - \hat{f}(\boldsymbol{\theta}_i) + \hat{f}(\boldsymbol{\theta}_i) - y_i + y_i - \ell(\boldsymbol{\theta}_i) \\
&\quad + \ell(\boldsymbol{\theta}_i) - \ell(\boldsymbol{\theta}_t)| \\
&\leq \left|\hat{f}(\boldsymbol{\theta}_t) - \hat{f}(\boldsymbol{\theta}_i)\right| + \left|\hat{f}(\boldsymbol{\theta}_i) - y_i\right| + |y_i - \ell(\boldsymbol{\theta}_i)| \\
&\quad + |\ell(\boldsymbol{\theta}_i) - \ell(\boldsymbol{\theta}_t)| \\
&\leq 2\lambda \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_i\|_1 + \beta \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_i\|_1 + \left|\hat{f}(\boldsymbol{\theta}_i) - y_i\right| \\
&\quad + |\ell(\boldsymbol{\theta}_i) - y_i| \\
&= (2\lambda + \beta) \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_i\|_1 + |\ell(\boldsymbol{\theta}_i) - y_i| + \left|\hat{f}(\boldsymbol{\theta}_i) - y_i\right|.
\end{aligned} \tag{23}$$

D. Proof of Proposition 8

If $\boldsymbol{\theta}_l, \boldsymbol{\theta}_u \in \mathcal{D}$ and $\boldsymbol{\theta}_l \leq \boldsymbol{\theta}_t \leq \boldsymbol{\theta}_u$, the following conditions hold as $\ell(\boldsymbol{\theta}_l) \leq \ell(\boldsymbol{\theta}_t) < \ell(\boldsymbol{\theta}_u)$, $\hat{f}(\boldsymbol{\theta}_l) \leq \hat{f}(\boldsymbol{\theta}_t) < \hat{f}(\boldsymbol{\theta}_u)$. Then the prediction error of $\hat{f}$ at $\boldsymbol{\theta}_t$ is bounded by

$$\begin{aligned}
&\left|\hat{f}(\boldsymbol{\theta}_t) - \ell(\boldsymbol{\theta}_t)\right| \\
&\leq \max\left\{\left|\hat{f}(\boldsymbol{\theta}_l) - \ell(\boldsymbol{\theta}_u)\right|, \left|\hat{f}(\boldsymbol{\theta}_u) - \ell(\boldsymbol{\theta}_l)\right|\right\}
\end{aligned} \tag{24}$$

where

$$\begin{aligned}
\left|\hat{f}(\boldsymbol{\theta}_l) - \ell(\boldsymbol{\theta}_u)\right| &= \left|\hat{f}(\boldsymbol{\theta}_l) - y_l + y_l - y_u + y_u - \ell(\boldsymbol{\theta}_u)\right| \\
&\leq |y_l - y_u| + |\hat{f}(\boldsymbol{\theta}_l) - y_l| + |\ell(\boldsymbol{\theta}_u) - y_u|
\end{aligned} \tag{25}$$

$$\begin{aligned}
\left|\hat{f}(\boldsymbol{\theta}_u) - \ell(\boldsymbol{\theta}_l)\right| &= \left|\hat{f}(\boldsymbol{\theta}_u) - y_u + y_u - y_l + y_l - \ell(\boldsymbol{\theta}_l)\right| \\
&\leq |y_l - y_u| + |\hat{f}(\boldsymbol{\theta}_u) - y_u| + |\ell(\boldsymbol{\theta}_l) - y_l|
\end{aligned} \tag{26}$$

Therefore, we have

$$\begin{aligned}
&\left|\hat{f}(\boldsymbol{\theta}_t) - \ell(\boldsymbol{\theta}_t)\right| \\
&\leq \max\left\{\left|\hat{f}(\boldsymbol{\theta}_l) - \ell(\boldsymbol{\theta}_u)\right|, \left|\hat{f}(\boldsymbol{\theta}_u) - \ell(\boldsymbol{\theta}_l)\right|\right\} \\
&\leq |y_l - y_u| + \max\Big\{\underbrace{|\ell(\boldsymbol{\theta}_u) - y_u|}_{\text{measurement error}} + \underbrace{|\hat{f}(\boldsymbol{\theta}_l) - y_l|}_{\text{fitting error}}, \\
&\qquad \underbrace{|\ell(\boldsymbol{\theta}_l) - y_l|}_{\text{measurement error}} + \underbrace{|\hat{f}(\boldsymbol{\theta}_u) - y_u|}_{\text{fitting error}}\Big\}.
\end{aligned} \tag{27}$$

REFERENCES

[1] E. Axell, G. Leus, E. G. Larsson, and H. V. Poor, "Spectrum sensing for cognitive radio: State-of-the-art and recent advances," *IEEE Signal Process. Mag.*, vol. 29, no. 3, pp. 101–116, May 2012.

[2] S. Yang, W. Yi, and A. Jakobsson, "Multitarget detection strategy for distributed MIMO radar with widely separated antennas," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–16, 2022.

[3] Y. Zhao, J. K. Nielsen, M. G. Christensen, and J. Chen, "Model-based voice activity detection in wireless acoustic sensor networks," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 425–429.

[4] M. Zhou et al., "Epileptic seizure detection based on EEG signals and CNN," *Front. Neuroinf.*, vol. 12, 2018, Art. no. 95.

[5] G. Ganesan and Y. Li, "Cooperative spectrum sensing in cognitive radio, part II: Multiuser networks," *IEEE Trans. Wireless Commun.*, vol. 6, no. 6, pp. 2214–2222, Jun. 2007.

[6] S. Atapattu, C. Tellambura, and H. Jiang, "Energy detection based cooperative spectrum sensing in cognitive radio networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 4, pp. 1232–1241, Apr. 2011.

[7] M. Park and H. Oh, "Cooperative information-driven source search and estimation for multiple agents," *Inf. Fusion*, vol. 54, pp. 72–84, Feb. 2020.

[8] R. Zhang, J. Zhang, Y. Zhang, and C. Zhang, "Secure crowdsourcing-based cooperative spectrum sensing," in *Proc. IEEE INFOCOM*, 2013, pp. 2526–2534.

[9] A. A. Khan, M. H. Rehmani, and A. Rachedi, "Cognitive-radio-based internet of things: Applications, architectures, spectrum related functionalities, and future research directions," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 17–25, Jun. 2017.

[10] Q. Wu, W. Wang, Z. Li, B. Zhou, Y. Huang, and X. Wang, "Spectrum-chain: A disruptive dynamic spectrum-sharing framework for 6G," *Sci. China Inf. Sci.*, vol. 66, no. 3, 2023, Art. no. 130302.

[11] H. D. Griffiths and C. J. Baker, *An Introduction to Passive Radar.* Norwood, MA, USA: Artech House, 2022.

[12] I. Potamitis and E. Fishler, "Microphone array voice activity detection and noise suppression using wideband generalized likelihood ratio," in *Proc. INTERSPEECH*, 2003, pp. 525–528.

[13] M. Cheng, W. Wang, Y. Zhang, X. Qin, and M. Li, "Target-speaker voice activity detection via sequence-to-sequence prediction," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2023, pp. 1–5.

[14] J. Xu, S. Mitra, C. Van Hoof, R. F. Yazicioglu, and K. A. Makinwa, "Active electrodes for wearable EEG acquisition: Review and electronics design methodology," *IEEE Rev. Biomed. Eng.*, vol. 10, pp. 187–198, 2017.

[15] Y. Zhang, Y. Guo, P. Yang, W. Chen, and B. Lo, "Epilepsy seizure prediction on EEG using common spatial pattern and convolutional neural network," *IEEE J. Biomed. Health Inform.*, vol. 24, no. 2, pp. 465–474, Feb. 2020.

[16] H. Urkowitz, "Energy detection of unknown deterministic signals," *Proc. IEEE*, vol. 55, no. 4, pp. 523–531, Apr. 1967.

[17] P. K. Varshney, *Distributed Detection and Data Fusion.* New York, NY, USA: Springer Sci. & Bus. Media, 2012.

[18] R. Tandra and A. Sahai, "SNR walls for signal detection," *IEEE J. Sel. Topics Signal Process.*, vol. 2, no. 1, pp. 4–17, Feb. 2008.

[19] A. Ghasemi, E. S. Sousa, "Collaborative spectrum sensing for opportunistic access in fading environments," in *Proc. 1st IEEE Int. Symp. New Frontiers Dynamic Spectr. Access Netw., (DySPAN)*, Piscataway, NJ, USA: IEEE Press, 2005, pp. 131–136.

[20] R. Zhang, T. J. Lim, Y.-C. Liang, and Y. Zeng, "Multi-antenna based spectrum sensing for cognitive radios: A GLRT approach," *IEEE Trans. Commun.*, vol. 58, no. 1, pp. 84–88, Jan. 2010.

[21] L. Huang, J. Fang, K. Liu, H. C. So, and H. Li, "An eigenvalue-moment-ratio approach to blind spectrum sensing for cognitive radio under sample-starving environment," *IEEE Trans. Veh. Technol.*, vol. 64, no. 8, pp. 3465–3480, Aug. 2015.

[22] Y. Zeng and Y.-C. Liang, "Eigenvalue-based spectrum sensing algorithms for cognitive radio," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1784–1793, Jun. 2009.

[23] A. Taherpour, M. Nasiri-Kenari, and S. Gazor, "Multiple antenna spectrum sensing in cognitive radios," *IEEE Trans. Wireless Commun.*, vol. 9, no. 2, pp. 814–823, Feb. 2010.

[24] D. Ramirez, G. Vazquez-Vilar, R. Lopez-Valcarce, J. Via, and I. Santamaria, "Detection of rank-p signals in cognitive radio networks with uncalibrated multiple antennas," *IEEE Trans. Signal Process.*, vol. 59, no. 8, pp. 3764–3774, Aug. 2011.

[25] Y. Lu, P. Zhu, D. Wang, and M. Fattouche, "Machine learning techniques with probability vector for cooperative spectrum sensing in cognitive radio networks," 2016 IEEE Wireless Commun. and Netw. Conf., 2016, Piscataway, NJ, USA: IEEE Press, pp. 1–6.

[26] Y. Zhang, Q. Wu, and M. R. Shikh-Bahaei, "On ensemble learning-based secure fusion strategy for robust cooperative sensing in full-duplex cognitive radio networks," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6086–6100, Oct. 2020.

[27] W. Lee, M. Kim, and D.-H. Cho, "Deep cooperative sensing: Cooperative spectrum sensing based on convolutional neural networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 3005–3009, Mar. 2019.

[28] J. H. Bae and M. Kim, "Performance improvement of cooperative spectrum sensing based on dequantization neural networks," *IEEE Wireless Commun. Lett.*, vol. 13, no. 5, pp. 1354–1358, May 2024.

[29] A. K. Dutta et al., "Deep learning-based multi-head self-attention model for human epilepsy identification from EEG signal for biomedical traits," *Multimedia Tools Appl.*, vol. 83, pp. 1–23, Mar. 2024.

[30] Y. Zeng, Y.-C. Liang, A. T. Hoang, and R. Zhang, "A review on spectrum sensing for cognitive radio: Challenges and solutions," *EURASIP J. Adv. Signal Process.*, no. 1, pp. 1–15, Jan. 2010.

[31] J. Xie, C. Liu, Y.-C. Liang, and J. Fang, "Activity pattern aware spectrum sensing: A CNN-based deep learning approach," *IEEE Commun. Lett.*, vol. 23, no. 6, pp. 1025–1028, Jun. 2019.

[32] D. Janu, K. Singh, and S. Kumar, "Machine learning for cooperative spectrum sensing and sharing: A survey," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 1, 2022, Art. no. e4352.

[33] A. Mariani, A. Giorgetti, and M. Chiani, "Effects of noise power estimation on energy detection for cognitive radio applications," *IEEE Trans. Commun.*, vol. 59, no. 12, pp. 3410–3420, Dec. 2011.

[34] A. Patel, H. Ram, A. K. Jagannatham, and P. K. Varshney, "Robust cooperative spectrum sensing for MIMO cognitive radio networks under CSI uncertainty," *IEEE Trans. Signal Process.*, vol. 66, no. 1, pp. 18–33, Jan. 2018.

[35] L. Zhou, W. Pu, M-y You, R. Zhang, and Q. Shi, "Joint optimization of UAV deployment and directional antenna orientation for multi-UAV cooperative sensing," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 1–5.

[36] R. Zhou, W. Pu, L. Zhao, M.-Y. You, Q. Shi, and S. Theodoridis, "Cooperative sensing via matrix factorization of the partially received sample covariance matrix," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP),* 2024, pp. 8881–8885.

[37] W. K. Hastings, *Monte Carlo Sampling Methods Using Markov Chains and Their Applications.* Oxford, U.K.: Oxford Univ. Press, 1970.

[38] Y. Zhao and D. Shrestha, "Uncertainty in position estimation using machine learning," in *Proc. Int. Conf. Indoor Positioning Indoor Navigation (IPIN),* Piscataway, NJ, USA: IEEE Press, 2021, pp. 1–7.

[39] S. Bartoletti, et al., "Uncertainty quantification of 5G positioning as a location data analytics function," in *Proc. Joint Eur. Conf. Netw. Commun. & 6G Summit (EuCNC/6G Summit)*, Piscataway, NJ, USA: IEEE Press, 2022, pp. 255–260.

[40] H. V. Habi, H. Messer, and Y. Bresler, "Learning to bound: A generative Cramér-Rao bound," *IEEE Trans. Signal Process.*, vol. 71, pp. 1216–1231, 2023.

[41] T. Diskin, Y. C. Eldar, and A. Wiesel, "Learning to estimate without bias," *IEEE Trans. Signal Process.*, vol. 71, pp. 2162–2171, 2023.

[42] A. Agresti and B. A. Coull, "Approximate is better than "exact" for interval estimation of binomial proportions," *Amer. Statist.*, vol. 52, no. 2, pp. 119–126, 1998.

[43] W. Feller, *An Introduction to Probability Theory and Its Applications.* New York, NY, USA: Wiley, 1991, vol. 81.

[44] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012.

[45] A. De La Fuente, N. Bing, I. Hoeschele, and P. Mendes, "Discovery of meaningful associations in genomic data using partial correlation coefficients," *Bioinformatics*, vol. 20, no. 18, pp. 3565–3574, 2004.

[46] R. Vallat, "Pingouin: Statistics in Python," *J. Open Source Softw*, vol. 3, no. 31, 2018, Art. no. 1026.

[47] J. Sill and Y. Abu-Mostafa, "Monotonicity hints," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 9, May 1997.

[48] A. Gupta, N. Shukla, L. Marla, A. Kolbeinsson, and K. Yellepeddi, "How to incorporate monotonicity in deep networks while preserving flexibility?" 2019, *arXiv:1909.10662*.

[49] X. Liu, X. Han, N. Zhang, and Q. Liu, "Certified monotonic neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 15427–15438.

[50] S. You, D. Ding, K. Canini, J. Pfeifer, and M. Gupta, "Deep lattice networks and partial monotonic functions," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[51] J. Sill, "Monotonic networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 10, 1997.

[52] A. Wehenkel and G. Louppe, "Unconstrained monotonic neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[53] N. Nolte, O. Kitouni, and M. Williams, "Expressive monotonic neural networks," in *Proc. 11th Int. Conf. Learn. Representations*, 2023.

[54] C. Anil, J. Lucas, and R. Grosse, "Sorting out Lipschitz function approximation," in *Proc. Int. Conf. Mach. Learn.,* PMLR, 2019, pp. 291–301.

[55] A. Abuduweili and C. Liu, "Estimating neural network robustness via Lipschitz constant and architecture sensitivity," 2024, *arXiv:2410.23382*.

[56] J. Ma and Y. G. Li, "Soft combination and detection for cooperative spectrum sensing in cognitive radio networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 11, pp. 4502–4507, Nov. 2008.

[57] A. Bollig, C. Disch, M. Arts, and R. Mathar, "Snr walls in eigenvalue-based spectrum sensing," *EURASIP J. Wireless Commun. Netw.*, vol. 2017, pp. 1–10, Jun. 2017.

[58] D. P. Kingma, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[59] M. D. Buhmann, "Radial basis functions," *Acta Numerica*, vol. 9, pp. 1–38, Jan. 2000.

[60] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: A survey," *Int. J. Mach. Learn. Cybern.*, vol. 2, pp. 107–122, May 2011.

[61] R. Zhou, W. Pu, L. Zhao, M.-Y. You, Q. Shi, and S. Theodoridis, "A matrix-factorization-error-ratio approach to cooperative sensing in non-ideal communication environment," *IEEE Trans. Signal Process.*, vol. 72, pp. 3851–3864, 2024.

[62] The MathWorks, Inc., *Communications Toolbox*, Natick, Massachusetts, United States. 2022. Accessed: Mar. 09, 2022. [Online]. Available: https://www.mathworks.com/help/comm/

[63] H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree, "Regularisation of neural networks by enforcing Lipschitz continuity," *Mach. Learn.*, vol. 110, pp. 393–416, Dec. 2021.

[64] O. Kitouni, N. Nolte, and M. Williams, "Robust and provably monotonic networks," *Mach. Learn.: Sci. Technol.*, vol. 4, no. 3, 2023, Art. no. 035020.

Rui Zhou (Member, IEEE) received the B.Eng. degree in information engineering from the Southeast University, Nanjing, China, in 2017, and the Ph.D. degree from The Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2021. Currently, he is a Research Scientist with Shenzhen Research Institute of Big Data and an Adjunct Assistant Professor with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China. His research interests include optimization algorithms, statistical signal processing, machine learning, and financial engineering.

Wenqiang Pu received the B.S. and Ph.D. degrees in electrical engineering from Xidian University, Xi'an, China, in 2013 and 2018, respectively. From 2019 to 2020, he was a Postdoctoral Associate with the School of Science and Engineering, The Chinese University of Hong Kong (Shenzhen). Currently, he is a Research Scientist with Shenzhen Research Institute of Big Data. His research interests include signal processing and optimization algorithms. His coauthored paper received Best Student Paper Award from IEEE SAM 2024. He serves as an Associate Editor of IEEE SIGNAL PROCESSING LETTERS.

Ming-Yi You received the B.S. and Ph.D. degrees in mechanical engineering from Shanghai Jiao Tong University, Shanghai, in 2006 and 2012, respectively. He was invited to visit the M. S. Wu Manufacturing Research Center, University of Michigan, from 2007 to 2008. Currently, he is a Senior Expert with the No. 36 Research Institute of CETC and leading a research group in radio direction finding and localization. He has published over 50 papers and coauthored the book *Radio Direction Finding: Theory and Practice*. His research interests include condition-based maintenance, radio direction finding, wireless localization, and multiobject tracking.

Qingjiang Shi (Member, IEEE) received the Ph.D. degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2011. From 2009 to 2010, he visited Prof. Z.-Q. (Tom) Luo's Research Group with the University of Minnesota, Twin Cities. In 2011, he worked as a Research Scientist with Bell Labs, China. In 2012, he was with the School of Information and Science Technology, Zhejiang Sci-Tech University. From 2016 to 2017, he worked as a Research Fellow with Iowa State University, USA. Since 2018, he has been with the School of Software Engineering, Tongji University, where he is currently a Full Professor. He is also with Shenzhen Research Institute of Big Data. His interests lie in algorithm design and analysis with applications in machine learning, signal processing, and wireless networks. So far, he has published more than 80 IEEE journals and filed about 40 national patents. He was an Associate Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING. He was the recipient of the IEEE Signal Processing Society Best Paper Award in 2022, the Huawei Technical Cooperation Achievement Transformation Award (2nd Prize) in 2022, the Huawei Outstanding Technical Achievement Award in 2021, the Golden Medal at the 46th International Exhibition of Inventions of Geneva in 2018, the First Prize of Science and Technology Award from China Institute of Communications in 2017, the National Excellent Doctoral Dissertation Nomination Award in 2013, the Shanghai Excellent Doctoral Dissertation Award in 2012, and the Best Paper Award from the IEEE PIMRC'09 conference.