Detecting Adversarial Spectrum Attacks via Distance to Decision Boundary Statistics

Wenwei Zhao*, Xiaowen Li†, Shangqing Zhao‡, Jie Xu§, Yao Liu†, Zhuo Lu*
*Department of Electrical Engineering, University of South Florida
†Department of Computer Science and Engineering, University of South Florida
‡School of Computer Science, University of Oklahoma
§Department of Electrical and Computer Engineering, University of Miami

***Abstract*—Machine learning has been adopted for efficient cooperative spectrum sensing. However, it incurs an additional security risk due to attacks leveraging adversarial machine learning to create malicious spectrum sensing values to deceive the fusion center, called adversarial spectrum attacks. In this paper, we propose an efficient framework for detecting adversarial spectrum attacks. Our design leverages the concept of the distance to the decision boundary (DDB) observed at the fusion center and compares the training and testing DDB distributions to identify adversarial spectrum attacks. We create a computationally efficient way to compute the DDB for machine learning based spectrum sensing systems. Experimental results based on realistic spectrum data show that our method, under typical settings, achieves a high detection rate of up to 99% and maintains a low false alarm rate of less than 1%. In addition, our method to compute the DDB based on spectrum data achieves 54%–64% improvements in computational efficiency over existing distance calculation methods. The proposed DDB-based detection framework offers a practical and efficient solution for identifying malicious sensing values created by adversarial spectrum attacks.**

***Index Terms*—Spectrum sensing, adversarial machine learning, attack detection**

I. Introduction

With the rapid development of wireless communication technology, spectrum resources are strained [1], [2]. Cooperative spectrum sensing can help improve the utilization of these resources by allowing multiple nodes to collect sensing data and report it to the fusion center [3], [4]. The fusion center then uses the sensing data to determine whether a primary channel is occupied or not such that unoccupied channels may be utilized while minimizing interference to primary users on occupied channels. However, it is possible for attackers to manipulate the sensing data in an attempt to deceive the fusion center into making incorrect decisions about channel availability. Attacks against cooperative spectrum sensing can take various forms, depending on the specific design of the system and the goals of the attacker. For example, an attacker could try to manipulate the sensing process by altering the signals received by the nodes [5], or by introducing false signals into the network [6]. An attacker could also try to interfere with the communication between nodes, in an attempt to disrupt the cooperative sensing process [7]. These attacks can have serious consequences, including disruption to primary user communication and low spectrum utilization.

The proliferation of machine learning techniques has facilitated some recent works [8], [9] to utilize machine learning for automatically learning features from data to effectively accomplish spectrum sensing tasks. Machine learning-based defenses [10], [11] have been proposed to counter different spectrum sensing attacks. However, this technological advancement has also presented new challenges as attackers have devised ways to exploit vulnerabilities in these systems. Recent studies [5], [12] have proposed to launch adversarial attacks against fusion center models from the perspective of machine learning. An adversarial attack is a type of attack on a machine learning model in which the attacker intentionally provides input to the model that is designed to cause the model to make a mistake [13] with minimum data change. It has been shown [5] that such attacks can effectively beat traditional methods that defend against spectrum data falsification. We call such attacks *adversarial spectrum attacks*.

To protect cooperative spectrum sensing against adversarial spectrum attacks, several methods have been proposed recently in the literature. The work in [14] proposes an effective defense based on autoencoder against adversarial attacks without affecting the performance of the model, but it runs on static datasets and does not account for any data variations or dynamics commonly existing in wireless environments. The work in [5] provides a method to mitigate harmful samples based on limiting the influence of each individual node on the fusion center's decision, but incurs a high computational cost and degrades the underlying sensing performance with the presence of an attack.

Due to the limitations of current approaches in spectrum sensing scenarios, our focus in this paper is on creating an effective defense to detect the presence of adversarial machine learning against cooperative spectrum sensing. The essential idea of our defense is to leverage the concept of the distance to the decision boundary (DDB) [15]. Decision boundaries are the boundaries that separate the different classes of data. The decision boundary in machine learning is determined during the training phase when the model learns to distinguish between the different classes based on the input features and their corresponding labels. Generally, the attackers aim to create subtle perturbations to deceive the classifier, which makes the data closer to a decision boundary. Based on this observation, it is possible to build a defense based on the

DDB statistics of sensing data over time. Even though the adversarial spectrum attacker can minimize the sensing data change to flip the fusion center's decision, the resulting data will incur a different DDB compared with the original data. Thus, we can use a distance statistic metric to measure the similarity between the original training data and the testing data to indicate the presence of an adversarial spectrum attack at the fusion center. There are two major components in our proposed detection mechanism: i) computing the DDB for an input of spectrum sensing data and ii) forming a DDB statistic over time to measure the similarity.

Finding and computing the DDB is essential in our approach. Typically, we need to determine the exact location of the decision boundary. However, in deep neural networks, locating the decision boundary can be challenging [16], [17]. The DeepFool method [18] proposes a way to compute the DDB based on a given model by iteratively perturbing input samples and checking the classifier's output until samples are misclassified. Other methods like Broyden-Fletcher-Goldfarb-Shanno (LBFGS) [19], the Carlini and Wagner method (C&W) [20], and the Decoupled Direction and Norm (DDN) [21] all aim to generate adversarial examples using different approximations to find the shortest distance. They usually require a substantial number of iterations and gradient calculations, therefore incurring a high computational cost for the fusion center to make a timely decision toward efficiently using spectrum resources. We propose to find the shortest distance by doing a binary search for data points along the direction perpendicular to a linear decision surface approximation for spectrum sensing data until we find the data point lies on the decision boundary. This not only mitigates the computational complexity of the method but also makes the finding of the DDB more straightforward for spectrum data classification.

After we get the DDBs in a sequence of spectrum sensing data inputs, we adopt the Kolmogorov-Smirnov test [22] to compare the DDB distribution of the data inputs with that of the training data. We collect realistic spectrum data and use experiments to show that the proposed detection is able to effectively detect the presence of adversarial spectrum attacks with typically 97% – 99% detection rates while maintaining low false alarms no more than 1% under the cooperative spectrum sensing scenarios with various settings. Our main contributions to this paper are as follows.

In summary, our major contributions are as follows. 1) We propose an attack detection framework that leverages the metric of DDB to detect the presence of adversarial spectrum attacks. 2) We create a new method to locate the decision boundary and compute the DDB in spectrum sensing applications. The new method is more computationally efficient than existing methods adopted in the machine learning community. 3) The experimental results based on realistic spectrum data illustrate the efficiency and effectiveness of our proposed DDB-based attack detection compared to existing approaches.

The rest of the paper is organized as follows. In Section II, we present the system model and the threat model in the paper. In Section III, we describe in detail the calculation of DDB and how we leverage the DDB to detect the presence of adversarial spectrum attacks. Next, we conduct experimental evaluations and discuss the results in Section IV. Finally, we review the related work in Section V and conclude this paper in Section VI.

II. System and Threat Models

In this section, we describe the system models, threat models and state our research problem.

A. System Model

We consider a cooperative spectrum sensing wireless network, in which there are n sensing nodes and one fusion center. At the i-timeslot, each sensing node first senses the energy level of the wireless channel. We denote all sensed values by a column vector $\mathbf{x}_i = [x_{i,1}, x_{i,2}, ..., x_{i,n}]^\top \in \mathbb{R}^{n\times 1}$, where $x_{i,j}$ $(j \in [1,n])$ is the j-th node's sensed energy value; and $\mathbb{R}^{n\times 1}$ denotes the n-dimensional real space. Then, all sensing nodes report $\mathbf{x}_i$ to a sensing data classifier f at the fusion center. The classifier f first uses two prediction functions $f_1 : \mathbb{R}^{n\times 1} \to \mathbb{R}$ and $f_0 : \mathbb{R}^{n\times 1} \to \mathbb{R}$ to compute the prediction scores for labels 1 and 0, denoting the channel available and unavailable, respectively. Then, f outputs the label with the higher prediction score as the sensing decision y_i for the i-th timeslot.

B. Threat Model

Due to the distributed nature of cooperative spectrum sensing, the fusion center's decision relies on the individual node's report. However, it is not always guaranteed [4] that each node in the network will not be compromised or behave in a selfish or malicious way. There are existing efforts in the literature on studying various attack and defense strategies in cooperative spectrum sensing. Such attacks are usually called Spectrum Sensing Data Falsification (SSDF) attacks [23]. Traditionally, to detect the presence of SSDF, there are several major design categories: (i) reputation based detection [24], [25], (ii) machine learning based detection [26], [27], (iii) cross-correlation based detection [28].

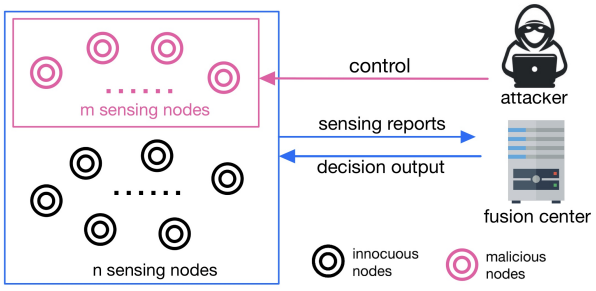

Fig. 1. Cooperative sensing scenario with malicious nodes controlled by an attacker.

Recently, due to the advancement of adversarial machine learning, a new type of attack [5] has been proposed against cooperative spectrum sensing. For example, when an attacker can control $m < n$ nodes in the network [5] (without loss of

generality, assuming nodes 1-m are compromised), as shown in Fig. 1, the attacker can consider the decision function f of the fusion center as a black box model with sensing data $\mathbf{x}_i$ as the input and decision y_i as the output. In this way, the attacker can observe a series of inputs and outputs in the network and build its own machine learning model to predict the decision of the fusion center. Then, the attacker can create adversarial examples at the m nodes that it controls [29] as falsified sensing reports to fool the fusion center to make an incorrect decision.

Mathematically, the attacker can create a new falsified data vector $\mathbf{x}_i' = \mathbf{x}_i + \delta_i$ by building the surrogate model S_i at timeslot i and find the perturbation vector δ_i that satisfies:

$$\text{Objective:} \quad \arg\min_{\delta_i \in \mathbb{R}^n} \|\delta_i\|_2 \tag{1}$$

$$\text{s.t.} \quad S_i(\mathbf{x}_i') \neq S_i(\mathbf{x}_i) \tag{2}$$

It has been shown in [5], [30] that such attacks can greatly degrade the performance of cooperative spectrum sensing even under traditional defense methods. We call such attacks leveraging adversarial machine learning to create falsified sensing data as *adversarial spectrum attacks*.

C. Problem Statement

The adversarial spectrum attack is an emerging security threat against cooperative spectrum sensing. Yet, there is no systematic study to detect such an attack in a cooperative sensing network. Some existing methods are limited in certain application scenarios and cannot be directly applied. For example, the autoencoder-based defense method [14] does not take into account any data variations or dynamics in wireless environments, and the randomized smoothing [31] may decrease a model's prediction accuracy. The efforts have also been made in [5] to mitigate the impact of adversarial spectrum attacks by limiting the influence of individual nodes on the fusion center's global decision. However, this leads to a non-negligible penalty on the sensing performance.

As a result, we consider the problem of adversarial spectrum attacks in this paper. We aim to propose a new detection method to accurately detect the presence of adversarial spectrum attacks in a cooperative spectrum sensing network.

III. Detection Leveraging DDB

In this section, we first define the decision boundary and the DDB and briefly introduce existing methods to find the DDB. Then, we propose our method to find the DDB for cooperative spectrum sensing applications.

A. Decision Boundary and DDB

1) Definitions: Some current studies [15], [32] have pointed out that subtle changes in the data will lead to significant changes in the distance between data and the decision boundary of a decision model.

The decision boundary and the DDB are defined based on the prediction functions $\{f_y\}_{y\in\{0,1\}}$ in the classifier at the fusion center. The classification decision is given by

$$\widehat{y}(\mathbf{x}) = \arg\max_y f_y(\mathbf{x}). \tag{3}$$

The decision boundary $\mathcal{D} \subset \mathbb{R}^{n\times 1}$ for $y \in \{0,1\}$ is defined as

$$\mathcal{D} := \{\mathbf{x} | f_0(\mathbf{x}) = f_1(\mathbf{x})\}. \tag{4}$$

Then, for a given data vector $\mathbf{x}$, we use the ℓ_2-norm to define its distance to the decision boundary

$$d(\mathbf{x}) = \min_{\delta\in\mathbb{R}^n} \|\delta\|_2 \quad \text{s.t.} \quad f_0(\mathbf{x}+\delta) = f_1(\mathbf{x}+\delta). \tag{5}$$

2) Finding Decision Boundary and DDB: Recently, there have been increasing efforts in the machine learning community to propose methods to investigate the decision boundary of a neural network model [15], [17], [33] generate particularly small adversarial perturbations for the data that keep it at the decision boundary approximation and considered adversarial examples as a type of DDB. Commonly used methods to find the DDB include DeepFool [18], Limited memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) [19], Carlini and Wagner (C&W) [20] methods.

DeepFool: The work of [15] takes the approximation in [34] based on DeepFool to find the distance of a point to the decision boundary. To compute the distance $d(\mathbf{x})$ in (4), the closed-form of perturbation $\delta_{\{k\}}$ at the k^{th} iteration can be written as

$$\delta_{\{k\}}(\mathbf{x}) = \frac{|f_0(\mathbf{x}_{\{k\}}) - f_1(\mathbf{x}_{\{k\}})|}{\|\nabla f_0(\mathbf{x}_{\{k\}}) - \nabla f_1(\mathbf{x}_{\{k\}})\|_2^2} D_\nabla(\mathbf{x}_{\{k\}}) \tag{6}$$

where $D_\nabla(\mathbf{x}_{\{k\}}) = |\nabla f_0(\mathbf{x}_{\{k\}}) - \nabla f_1(\mathbf{x}_{\{k\}})|$ and $\mathbf{x}_{\{k+1\}} = \mathbf{x}_{\{k\}} + \delta_{\{k\}}(\mathbf{x})$. When the iteration stops at the decision boundary after K iterations, and the perturbation is given by $\delta(\mathbf{x}) = \sum_{k=0}^{K-1} \delta_{\{k\}}(\mathbf{x})$, and we can get the DDB $d(\mathbf{x}) = \|\delta(\mathbf{x})\|_2$.

LBFGS: A box-constrained optimizer is used in [19] to minimize the perturbation and approximate (5) as

$$\min_{\delta\in\mathbb{R}^n} C\|\delta\|_2 + \text{loss}_f(\mathbf{x}+\delta, y_{\text{target}}) \tag{7}$$

$$\text{s.t.} -\mathbf{M} \leq \mathbf{x}+\delta \leq \mathbf{M}, \tag{8}$$

where $\text{loss}_f(\mathbf{x}+\delta, y_{\text{target}})$ is the loss function, y_{target} is the targeted label of the adversarial example, $-\mathbf{M}$ and $\mathbf{M}$ constrain the range of the adversarial example. A line search is applied to update the constant C to optimize the function.

C&W: [20] proposed to change variables by using the tanh function instead of using a box-constrained optimizer via alternating (5) with the following equation:

$$\min_{\delta\in\mathbb{R}^n} \|\delta\|_2 + CF(\mathbf{x}+\delta) \tag{9}$$

$$\text{s.t.} -\mathbf{M} \leq \mathbf{x}+\delta \leq \mathbf{M}, \tag{10}$$

where function $F(\mathbf{x}+\delta)$ is based on the best objective function and is defined as

$$F(\mathbf{x}+\delta) = \max(\max\{f(\mathbf{x}+\delta)_{y_1}\} - f(\mathbf{x}+\delta)_{y_0}, \kappa) \tag{11}$$

$$\text{s.t. } y_0, y_1 \in \{0,1\},\ y_0 \neq y_1 \tag{12}$$

with κ used to control the confidence in the occurrence of misclassification. Because of the box constraints in (10), C&W

introduced a new variable ρ and used tanh function [35] to rewrite the perturbation δ as

$$\delta = \frac{1}{2}(\tanh(\rho) + 1) - \mathbf{x} \tag{13}$$

It then applied the change-of-variables and optimize over ρ for (9).

B. Proposed Method to Compute DDB for Spectrum Sensing

1) Motivation: DeepFool, LBFGS, and C&W all adopt an iterative procedure to estimate the distance between a given data point and the decision boundary but have high computational complexity. In order to make DDB-based attack detection more effective, we propose a simpler and faster method to calculate the DDB. Our observation is that in spectrum sensing, when the signal energy level increases, the wireless channel is more likely considered occupied; similarly, when the energy level decreases, the channel is more likely to be available. Thus, adding or deducting the value of the energy level should always flip the decision of the fusion center. By increasing or decreasing the values in $\mathbf{x}_i$ at the i-th timeslot along a certain direction $\mathbf{u}$, we can find the shortest distance from the data point to the decision boundary of whether the channel is available or not. Fig. 2 gives illustrative examples to compare an iterative method with our idea of finding the DDB.

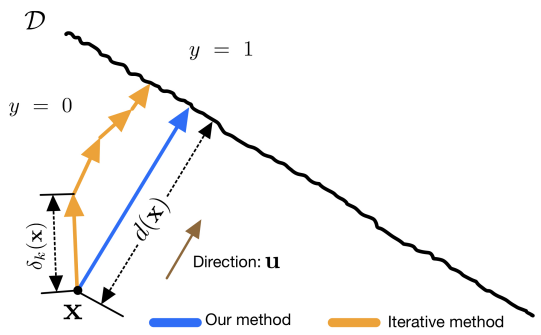

Fig. 2. Our idea to find the direction vs an iterative method to find the DDB.

Then, our goal becomes designing a computationally efficient method to find the direction along which we can estimate the DDB for spectrum sensing classification. To this end, we first notice that using machine learning for cooperative spectrum sensing is commonly motivated by the fact that sensing data can be unreliably collected and vary over time. Thus, machine learning becomes an efficient way to characterize such data without explicitly assuming particular sensing data models [5]. However, the advantage of assuming sensing data distributions is that an optimal solution in terms of maximizing a performance metric can be developed. When a machine learning classifier is sufficiently trained by the sensing data, it should achieve a performance close to the optimal solution. Hence, our idea is to use the theoretically optimal solution as a guide for estimating the direction that we should use to compute the DDB for a machine learning classifier.

2) Finding the Direction to Compute DDB: We adopt the theoretical framework of the likelihood ratio test (LRT) based on the Neyman-Pearson lemma [8] to establish the decision boundary between sensing output labels 1 (channel available) and 0 (not available). Once the decision boundary is established, we use the direction from $\mathbf{x}_i$ perpendicular to the theoretical boundary to estimate the shortest distance to the actual boundary in the classifier.

We assume that the fusion center has a full view of all signals from sensing nodes. Denote by $\mathbf{v}_i(t) \in \mathbb{C}^{n\times 1}$ the vector of the t-th received signal samples ($t \in \{1, , 2, \cdots, T\}$) at all nodes at the i-th timeslot, where $\mathbb{C}^{n\times 1}$ denotes the n-dimentional complex space. We can write $\mathbf{v}_i(t) = \mathbf{s}_i(t)+\eta_i(t)$, where $\mathbf{s}_i(t)$ is the deterministic complex signal sent by a primary user and $\eta_i(t)$ is the Gaussian noise; and

$$\mathbf{v}_i(t) = \begin{cases} \mathbf{s}_i(t) + \eta_i(t), & \text{channel unavailable} \\ \eta_i(t), & \text{channel available.} \end{cases} \tag{14}$$

3) Analysis and Design: The signal power sensed by node j at timeslot i can be denoted as $x_{ij} = \frac{1}{T}\sum_{t=1}^{T} |v_{ij}(t)|^2$, where $v_{ij}(t)$ is the j-th element in $\mathbf{v}_i(t)$. In cooperative spectrum sensing, the fusion center collects the signal power value instead of the signal samples from each node and makes a decision based on the fusion rule. Based on (14), the distributions of the signal power when the channel is available (hypothesis $\mathscr{H}_1$) and unavailable (hypothesis $\mathscr{H}_0$) are both Gamma distributions with same shape parameter T, but different scale parameters β_{ij}^0 and β_{ij}^1.

$$\mathscr{H}_0 \quad : \quad x_j \sim \Gamma(T, \beta_{ij}^0), \tag{15}$$

$$\mathscr{H}_1 \quad : \quad x_j \sim \Gamma(T, \beta_{ij}^1). \tag{16}$$

After mathematical manipulations, the probability density functions of (15) and (16) can be expressed as

$$p(x_j|\mathscr{H}_0) = \frac{x_j^{T-1} e^{-\frac{x_j}{\beta_{ij}^0}}}{(\beta_{ij}^0)^T \Gamma(T)}, \tag{17}$$

$$p(x_j|\mathscr{H}_1) = \frac{x_j^{T-1} e^{-\frac{x_j}{\beta_{ij}^1}}}{(\beta_{ij}^1)^T \Gamma(T)}. \tag{18}$$

Then, according to LRT, we can write

$$\Lambda = \frac{p(\mathbf{x}_i|\mathscr{H}_1)}{p(\mathbf{x}_i|\mathscr{H}_0)} = \frac{\prod_{j=1}^{n} p(x_{ij}|\mathscr{H}_1)}{\prod_{j=1}^{n} p(x_{ij}|\mathscr{H}_0)} \underset{\mathscr{H}_0}{\overset{\mathscr{H}_1}{\gtrless}} \gamma, \tag{19}$$

where γ is a given threshold for detection. After mathematical manipulations, (19) can be expressed as

$$\sum_{j=1}^{n} \left(\frac{1}{\beta_{ij}^0} - \frac{1}{\beta_{ij}^1}\right) x_{ij} \underset{\mathscr{H}_0}{\overset{\mathscr{H}_1}{\gtrless}} \ln(\gamma) - T\sum_{j=1}^{n} \ln\left(\frac{\beta_{ij}^0}{\beta_{ij}^1}\right) \tag{20}$$

It can be seen that the decision boundary of the hypotheses $\mathscr{H}_0$ and $\mathscr{H}_1$ is linear, and can be expressed as $\mathcal{D} = \{\mathbf{x} : \mathbf{w}^\top \mathbf{x} + b = 0\}$, where $\mathbf{w} = [(\frac{1}{\beta_1^0} - \frac{1}{\beta_1^1}), (\frac{1}{\beta_2^0} - \frac{1}{\beta_2^1}), ..., (\frac{1}{\beta_n^0} -$

Algorithm 1: Pseudocode of DDB Algorithm

Data: Sensing data $\mathbf{x_i}$, initial step length ϵ, direction $\mathbf{u}$, stop threshold ξ;
Result: DDB $\delta_\mathbf{i}$;

```
if y(x_i) = 0 then
    while y(x_i + ε u) = 0 do
        ε = 2ε;
    end
    x_l ← x_i, x_r ← x_i + ε u;
else if y(x_i) = 1 then
    while y(x_i − ε u) = 1 do
        ε = 2ε;
    end
    x_l ← x_i − ε u, x_r ← x_i;
else
    return "False";
end
repeat
    x' ← (x_l + x_r)/2;
    if y(x_i) = 0 then
        x_l ← x';
    else if y(x_i) = 1 then
        x_r ← x';
    else
        return "False";
    end
until |x_l − x_r| ≤ ξ;
return δ_i = |x' − x|;
```

$\frac{1}{\beta_n^1})]$, where $\beta_j^0 = \beta_{ij}^0$ and $\beta_j^1 = \beta_{ij}^1$ for all timeslots. Thus, for a given sensing vector $\mathbf{x_i}$, the vector $\mathbf{w}$ denotes the direction of its shortest path to the decision boundary $\mathcal{D}$, and its DDB can be given by

$$d(\mathbf{x}_i) = \min_{\delta \in \mathbb{R}^{n\times 1}} \|\delta\|_2 = \frac{|f(\mathbf{x}_i)|}{\|\mathbf{w}\|_2} \tag{21}$$

$$\text{s.t. } f_0(\mathbf{x}_i + \delta) = f_1(\mathbf{x}_i + \delta) \tag{22}$$

where f is the prediction function of the fusion center, f_0 and f_1 are the prediction scores for classes 0 and 1 of classifier f, respectively.

A binary search strategy can be used to efficiently solve (21). More specifically, to find the distance from the sensing vector $\mathbf{x}_i$ to the decision boundary $\mathcal{D}$, we create a new vector along the search direction $\mathbf{u} = -\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$ as $\mathbf{x}_i + \epsilon\mathbf{u}$, where ϵ is large enough such that the new vector flips the fusion center's decision. Then, the binary search is used to find the boundary point from $\mathbf{x}_i$ to $\mathbf{x}_i + \epsilon\mathbf{u}$ along the direction $\mathbf{u}$. The distance is then calculated as the Euclidean distance between $\mathbf{x}_i$ and the boundary point. The pseudocode of this algorithm is shown in Algorithm 1.

C. Detection Method

After computing the DDB, we can collect the DDB for each training sample during the training process and obtain the distribution of the DDB from training. This distribution will serve as the ground truth DDB distribution. Our intuition for attack detection is that the attacks based on adversarial machine learning always aim to minimize the data perturbation to just go across the decision boundary of a classifier. Thus, if we measure the DDB distribution of testing data under the adversarial spectrum attacks, it should be quite different from the training distribution. This becomes the basis for attack detection by comparing the training and testing DDB distributions.

As a result, we use the consistency of the distributions between the training and testing DDB sets to detect whether there exists an adversarial spectrum attack against the fusion center. We choose the Kolmogorov-Smirnov (K-S) test [22], which is commonly used as a nonparametric test to analyze whether two sets of data are different especially when the sample sizes in the two sets are relatively small.

Given a training DDB set with size a_1 and a testing DDB set with size a_2, we calculate their respectively Cumulative Distribution Functions (CDFs) $F_{\text{train},a_1}(\delta)$ and $F_{\text{test},a_2}(\delta)$. The maximum distance d_{KS} of the K-S statistic is $d_{\text{KS}} = \max_{\delta} |F_{\text{train},a_1}(\delta) - F_{\text{test},a_2}(\delta)|$. Consider the null and alternative hypotheses

$$\mathcal{H}_0 \quad : \quad x_{\text{train}} \text{ and } x_{\text{test}} \text{ are from same distribution} \tag{23}$$

$$\mathcal{H}_1 \quad : \quad x_{\text{train}} \text{ and } x_{\text{test}} \text{ are from different distributions} \tag{24}$$

where $\mathcal{H}_0$ indicates that there is no attack and $\mathcal{H}_1$ means that an attack with manipulating spectrum data. According to the K-S test, we can write the decision rule as

$$P_{\text{value}} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\lessgtr}} \alpha, \tag{25}$$

where

$$P_{\text{value}} = 2e^{-2d_{\text{KS}}^2 \frac{a_1 a_2}{a_1 + a_2}}, \tag{26}$$

and the decision threshold α is called the significance level.

D. Can Attack Bypass DDB-based Detection?

Our attack detection is to first compute the DDBs of spectrum sensing data vectors, then compare the DDB distributions to indicate the presence of an attack. The motivation behind the detection is that adversarial sensing data vectors manipulated by an attacker should always have very small distances to the decision boundary, but benign sending data vectors do not. Is it possible for an attacker to create adversarial spectrum sensing data while maintaining the same DDB distribution to evade our DDB-based detection method? In the following, we show that the attacker cannot create such attacks because of a lack of information.

In our threat model in Section II, the attacker can only access and control m out of n ($m < n$) nodes in the network. That means for sensing data vector $\mathbf{x_i} \in \mathbb{R}^{n\times 1}$ at timeslot i, only $[x_{i,1}, x_{i,2}, ..., x_{i,m}]^\top \in \mathbb{R}^{m\times 1}$ can be changed by the attacker and $[x_{i,m+1}, x_{i,m+2}, ..., x_{i,m+(n-m-1)}, x_{i,n}]^\top \in \mathbb{R}^{n-m\times 1}$ is always the same regardless attack or not. If the attacker wants to defeat the DDB-based defense, the attacker

should generate malicious data that will be classified into a different label instead of the original label, and at the same time, the DDB of the malicious data should have the same distribution as the ground truth.

Suppose that the attacker generates an adversarial sensing data vector with a target DDB of d_t from the vector to the decision boundary. According to (21), the objective function can written as

$$d_t = \frac{|\mathbf{w}^\top \mathbf{x}'_\mathbf{i} + b|}{\|\mathbf{w}\|_2} \tag{27}$$

where $\mathbf{x}'_\mathbf{i}$ is the adversarial spectrum data sensing vector, and (27) can be further expressed as

$$\sum_{j=1}^{m} w_j x'_{ij} = \pm d_t \|\mathbf{w}\|_2 - \sum_{j=m+1}^{n} w_j x_{ij} - b \tag{28}$$

To find such an adversarial vector $\mathbf{x}'_\mathbf{i}$, the attacker must know all the parameters in (28), and get the values of $[x_{i,m+1}, x_{i,m+2}, ..., x_{i,n}]$. However, without the information of remaining $n - m$ nodes, the attacker cannot get the $n - m$ values in vector $\mathbf{x_i}$ and $\mathbf{w}$. As a result, it is generally infeasible for the attacker to create an adversarial vector with a specifically targeted DDB.

IV. Experimental Evaluations

In this section, we present our experimental evaluations. We first introduce the experimental setup and spectrum data set collection, then discuss the evaluation results.

A. Evaluation Setups

1) Spectrum Sensing Scenario: We set up a cooperative sensing network consisting of the data fusion center and $n = 20$ sensing nodes, in which $m < 10$ nodes are compromised and try to launch adversarial attacks against the fusion center. We use realistic spectrum sensing data for the evaluation. In particular, the data includes realistic white space TV signal strengths collected by reliable RTL-SDR TV dongles. The signal strengths on the TV channels were collected simultaneously at 20 different locations (interior and exterior of a building or different floors in the building) at a university campus. The sensing process at each location measured the average signal power over a 30-second time period as the sensing result for each time slot, as required by the Federal Communications Commission [36].

2) Spectrum Data Fusion Center: We implement a multi-layer perceptron (MLP) classifier at the fusion center to classify the spectrum sensing data into benign or malicious. We use 20,000 spectrum sensing data vectors as training data to train the classifier, which can achieve a high sensing accuracy of 99.94% without attack. Then, in our experiment, we use additional 80,000 sensing data vectors to simulate the attacking scenario in which m malicious nodes aim to manipulate their corresponding entries in the sensing data vector to fool the fusion center. We note that when there is no attack, the classifier at the fusion center can always correctly classify each of the 80,000 sensing data vectors.

3) Attack Detection and Evaluation Metrics: We deploy the DDB-based detector at the fusion center to detect the malicious behavior of the m malicious nodes. We aim to evaluate how our proposed DDB calculation method compares with the DDB estimates by using DeepFool, LBFGS, and C&W in the literature. Once we obtain the DDBs from different calculation methods, we always use the K-S test to compare the DDB distributions. We use the detection rate and false alarm rate as the evaluation metrics for the detection performance. The detection rate is the probability that the detector indicates an attack in the presence of an attack. The false alarm rate is the probability that the detector indicates an attack, but there is no actual attack in the system.

4) Default Settings: During our evaluations, we adopt the following default settings and evaluate the scenario in the presence of the attack. The first step length of Binary Search is $\epsilon = 5$ and the threshold to stop searching is $\xi = 0.01$. The number of malicious nodes is $m = 7$ and the attacker uses the Fast Gradient Sign Method (FGSM) [37] to generate adversarial spectrum data vectors. In the K-S test, the group size a_2 for attack detection is 25; and the threshold α of P_{value} is set to 0.01, meaning that the distributions of ground truth data and test data are different at least with the 99% confidence level in the detection.

B. Evaluation Results

1) Impact of Test Group Size on Attack Detection: Generally, the more test data involved in the detection, the more accurate the attack detection should be. However, involving more data prolongs the attack detection time. Thus, we first evaluate the impact of the test group size a_2 on the attack detection performance. We choose 5, 10, 20, 25, 50, 80, 100, 200, and 400 as the group size to detect the presence of the adversarial spectrum attack.

Fig. 3 shows the detection rates of different methods as the group size increases from 5 to 400. First, we can see that improving the group size clearly increases the detection performance. Our proposed DDB method achieves approximately the same performance as existing methods including DeepFool, C&W, and LBFGS. The attack detection rate approaches 100% when the size of the group used for attack detection exceeds 20, indicating the relatively fast detection performance of 20 timeslot observation for each method. Fig. 4 shows the false alarm rates under the same conditions. We can see that all methods have low false alarm rates under 0.01.

Overall, the results from Figs. 3 and 4 show that the DDB-based attack detection achieves good detection rates with low false alarm rates. In addition, our proposed DDB calculation method leads to the approximately same performance as DeepFool, C&W, and LBFGS.

2) Impact of Attack Frequency: Generally, attackers may not keep launching attacks overall timeslots. We are interested in how the frequency of the attacks affects the results of our detection method. We define the metric of attack occurrence ratio (i.e., the probability that a timeslot will be attacked) to measure the attack intensity. Fig. 5 shows the detection rates

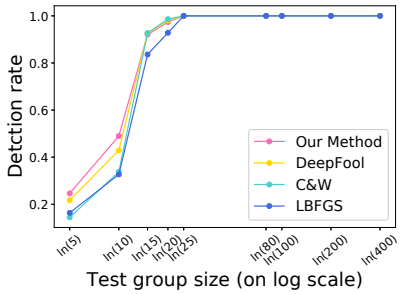

Fig. 3. Detection rates under the different test group sizes.

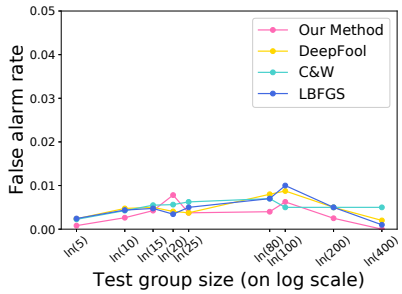

Fig. 4. False alarms under the different sizes of test data.

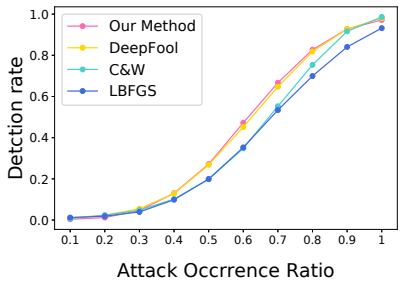

Fig. 5. Detection rates under different attack occurrence ratios.

TABLE I
DETECTION RATES UNDER DIFFERENT ATTACK OCCURRENCE RATIOS.

Methodology	Group Size	Attack Occurrence Ratio									
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Our Method	200	0.0150	0.3800	0.9600	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	100	0.0163	0.0950	0.4850	0.8763	0.9888	1.0000	1.0000	1.0000	1.0000	1.0000
	50	0.0075	0.0244	0.1338	0.4094	0.7438	0.9056	0.9831	0.9981	1.0000	1.0000
	25	0.0038	0.0119	0.0466	0.1303	0.2716	0.4722	0.6669	0.8272	0.9272	0.9741
	10	0.0044	0.0076	0.0143	0.0334	0.0574	0.1051	0.1743	0.2626	0.3706	0.4833
DeepFool	200	0.0375	0.4050	0.9650	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	100	0.0175	0.1063	0.5163	0.8988	0.9988	1.0000	1.0000	1.0000	1.0000	1.0000
	50	0.0156	0.0406	0.1719	0.4231	0.7363	0.9206	0.9856	0.9981	1.0000	1.0000
	25	0.0109	0.0234	0.0553	0.1284	0.2681	0.4516	0.6472	0.8184	0.9272	0.9803
	10	0.0066	0.0113	0.0191	0.0381	0.0608	0.1043	0.1593	0.2398	0.3253	0.4320
C&W	200	0.0600	0.3450	0.7875	0.9800	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	100	0.0250	0.1225	0.3875	0.7313	0.9338	0.9950	1.0000	1.0000	1.0000	1.0000
	50	0.0150	0.0531	0.1438	0.3119	0.5856	0.8038	0.9519	0.9931	0.9994	1.0000
	25	0.0069	0.0238	0.0478	0.1019	0.1981	0.3481	0.5525	0.7534	0.9163	0.9863
	10	0.0055	0.0058	0.0085	0.0161	0.0266	0.0479	0.0780	0.1338	0.2075	0.3304
LBFGS	200	0.0375	0.2750	0.8750	0.9950	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	100	0.0225	0.0763	0.3775	0.7725	0.9738	0.9975	1.0000	1.0000	1.0000	1.0000
	50	0.0200	0.0269	0.0925	0.2888	0.5725	0.7981	0.9375	0.9850	1.0000	1.0000
	25	0.0122	0.0188	0.0394	0.0988	0.1991	0.3528	0.5341	0.6994	0.8406	0.9281
	10	0.0071	0.0091	0.0171	0.0293	0.0465	0.0741	0.1141	0.1689	0.2468	0.3289

of the DDB-based methods with the attack occurrence ratios going from 0.1 to 1.

It can be seen from Fig. 5 that the detection rates of all four methods improve as the attack occurrence ratio becomes larger. The detection rate of each method is low when the attack occurrence ratio is low. In such a scenario, the fusion center's decision will not be substantially affected because the attack happens rarely. When the attack occurrence ratio increases to 1, the detection performance gradually approaches 100% for each method. All four methods achieve similar detection performance while our method holds slight advantages over DeepFool, C&W, and LBFGS.

We also measure the detection rates under different test group sizes in Table I. We can see from the table that a larger group size gives a better detection rate (but delays a detection decision). The detection rate in our method is 96% when the attack occurrence ratio is 30% with a group size of 200. But when the group size is 10, the detection rate is only 1.43%. This is due to the fact that the larger number of test data can more accurately reflect the distribution of data in the test group. As a result, we need to increase the test group size in order to detect a low-frequency attack.

3) Impact of Number of Malicious Nodes: We evaluate the impact of the number of malicious nodes m on the attack detection performance. In particular, we set $m = 3, 5, 7, 10$ and measure the attack detection rates in Fig. 6.

From Fig. 6, we observe that as m increases, its influence on the DDB becomes greater, leading to a more obvious DDB distribution difference and a larger detection rate. For example, we can see when $m = 3$, the detection rate of our method is 80%; and when $m = 10$, the detection rate becomes 98%. In addition, we also note that even DeepFool, C&W, and LBFGS have worse performance than our method when m is small and gradually catches up with the detection rate when m increases. For attackers, a large number of malicious nodes make it easier for attackers to change the value of x to generate adversarial example x', a significant change in x also means that the DDB changes are more significant and thus easier to detect.

4) Comparison of Time Complexity: The evaluation results show that our DDB method generally has the sample attack detection performance, in terms of detection rate and false alarm, with DeepFool, LBFGS, and C&W. We compare their run-time efficiency for the comparison of complexity. As all the methods are iteration-based, we use two metrics (i) the number of iterations required and (ii) the average time needed

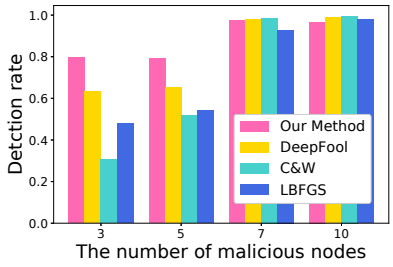

Fig. 6. Detection rates under the different numbers of malicious nodes.

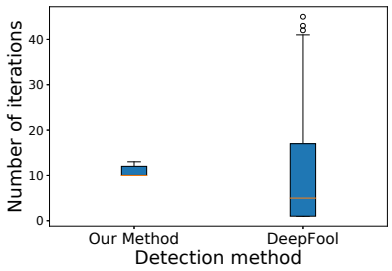

Fig. 7. Comparison of the number of iterations between our method and DeepFool.

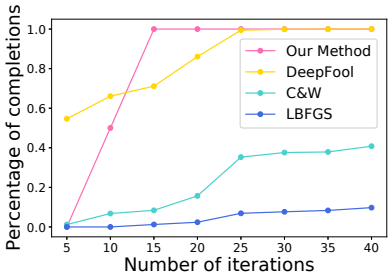

Fig. 8. Percentages of calculations that have been completed in different given numbers of iterations.

TABLE II
TIME COMPARISONS AMONG DIFFERENT METHODS.

Methodology	Time Consumption/Iteration
Our Method	2.41 ms
DeepFool	5.26 ms
C&W	6.71 ms
LBFGS	5.85 ms

for each iteration for the complexity evaluation. We implemented DeepFool, C&W, and LBFGS based on CleverHans V3.1.0 [38].

Fig. 7 boxplots the numbers of iterations of our method compared with DeepFool. We only compare these two methods as they complete with much fewer average iterations than LBFGS and C&W methods with hundreds or even thousands of iterations. From the figure, we can observe that DeepFool has an average number of iterations less than that of our method (8.75 vs 10.98). However, we can see from Fig. 7 that DeepFool, due to its interactive nature, has a large variance in the number of iterations. The cumulative distribution function of the number of iterations required for each method is shown in Fig. 8. It is seen from the figure that at the number of iterations of 5, DeepFool performs best with 54% calculations completed; while our method, C&W, and LBFGS do not have any completed calculations. When the number of iterations becomes 15, our method completes more than 99% of the calculations, leading to a better computational complexity than the other three methods (DeepFool: 71%, C&W: 8%, and LBFGS: 2%).

Table II shows the average time needed for each method to complete one iteration. We can see that our method has the shortest time consumption with 2.41 microseconds per iteration. Overall, to find the DDB for one sensing data vector, our method can save, on average, 54%, 64%, and 59% of the time for one iteration for DeepFool, C&W, and LBFGS, respectively.

5) Detection Performance under Attack Generation Methods: Next, we compare the performance of our defense strategies in different attack approaches that the attacker may use for adversarial sensing result generation. These methods are: Fast gradient sign method (FGSM) [37], Projected Gradient Descent (PGD) [32], DeepFool [18], Elastic-Net method (EN) [39] and L-BFGS [19] in Deep Neural Network, which is implemented based on CleverHans V2.1.0 [38]. The detection rate and false alarm of defending against these different attacks are shown in Fig. 9. DDB-statistic based detection method shows good performance for most attack methods. Even if the results show that our method performs differently under different attack methods, it achieves at least 80% detection rate and is overall better than the other three methods.

In theory, no matter which adversarial sample generation method is used, the trend of the data points will be close to the direction of the decision boundary. Our DDB-statistic based detection method can effectively detect attacks when facing different approaches adopted by the attacker, but the performance is still not ideal under some attack methods (e.g., EN). A potential reason can be that the EN method has a very low attack success rate of 8.67% when creating adversarial example x', which means that x' changes less compared to x, which makes it difficult to detect with the DDB method. The attack success rates of using FGSM, PGD, DeepFool, and LBFGS methods are 42.37%, 26.84%, 57.12%, and 31.53%, respectively.

6) Impact of Locations of Malicious Nodes: Although we test different numbers of malicious nodes in our experiment, we still need to consider whether a given number of malicious nodes from different locations will affect the effectiveness of the defense. We choose 4 different nodes among the total of 20 nodes to form 5 groups of differently located malicious nodes. The detection rates are shown in Fig. 10. It can be seen from Fig. 10 that malicious nodes in different locations will have different impacts on the attack detection performance. We find that malicious nodes have distinct attack success rates at different locations. The higher the attacker's success rate, the higher the attack detection rate. For example, If we place malicious nodes at Location Group 1, they have the highest attack success rate (thereby causing the most damage to the network) and the DDB-based attack detection rate is also the highest in this case.

In all methods shown in Fig. 10, our method leads to the highest detection rate in most of the location groups, showing that it is overall more efficient than other distance methods.

7) Impact of Different Detection Thresholds: We also evaluate the impact of setting the threshold α of P_{value} in the K-S test on the attack detection performance. We choose 5

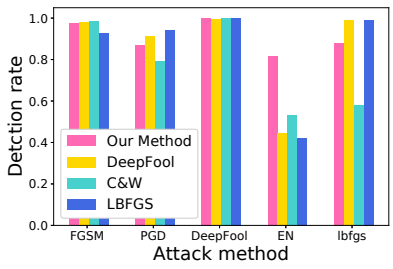

Fig. 9. The detection rates under different attack methods.

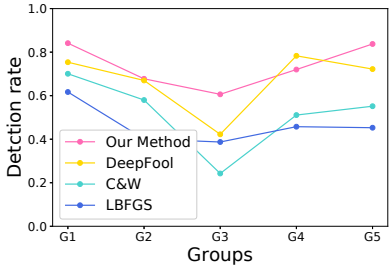

Fig. 10. The detection rates under the different locations of malicious nodes.

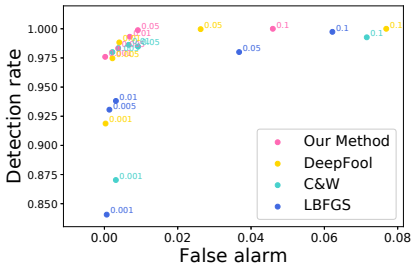

Fig. 11. The detection rates under the different thresholds in the K-S test.

values of α in the experiment and show the results, pairing both the detection rate and false alarm, in Fig. 11. It is noted from the figure that when $\alpha = 0.01$ (meaning that ground truth data and sensing data are different at least with the 99% confidence level), the detection rate and false alarm of our method are around 99.316% and 0.691%, respectively. And when $\alpha = 0.001$, our method and DeepFool have the lowest false alarm which is 0.0213%. Overall, we can see that our method has similar performance to other methods while being more computationally efficient.

V. Related Work

In this section, we summarize existing studies that are related to the work in this paper.

Attacks and defenses in cooperative spectrum sensing: Cooperative spectrum sensing is widely used in wireless communication systems, particularly in cognitive radio networks. Attacks against cooperative spectrum sensing can compromise the accuracy and integrity of spectrum sensing results and disrupt network operations. Some common types of attacks on cooperative spectrum sensing include: spectrum sensing data falsification (SSDF) [40], collusion attacks [41], and jamming attacks [42]. In this paper, our defense method mainly targets SSDF, in which malicious nodes deliberately provide false or misleading sensing data to manipulate the decision-making process [24], [40]. To counter SSDF attacking strategies, a lot of defense methods have been proposed in the literature. However, conventional studies [43] assumed certain prior knowledge of attacks. It has been shown in recent attack strategies [5], [14], [44] that leveraging adversarial machine learning can beat the conventional defense and poses a new challenge to secure cooperative spectrum sensing. Our study targets the recent adversarial machine learning based attacks and adopts the concept of DDB to efficiently detect the presence of the adversarial spectrum learning attack.

Combating adversarial examples: It is important to note that while methods have been proposed in the machine learning community to combat adversarial attacks in data classification applications, they may not necessarily work or be efficient in the spectrum sensing scenario. Generally, a machine learning model can be made more robust against small attack perturbations by training the model on a dataset that includes perturbed versions of the original data [37], [45], or by incorporating a regularization term in the model's loss function that penalizes large perturbations in the input data [46]. These methods are generally used to combat adversarial images by empirically setting a threshold of perturbations to the original data such that the perturbations do not affect the ground truth under human judgment. However, in cooperative spectrum sensing, there is no human judgment on any sensing data and there is no ground truth regarding what the sensing result should be if training data is modified with intentional perturbations. It becomes difficult to directly apply similar ideas to make the spectrum sensing data classification model robust against adversarial attacks. The DDB-statistic based detection method has shown its efficiency in detecting adversarial spectrum learning attacks.

Computing the DDB: The DeepFool method [18] proposed a way to compute the DDB based on a given model by iteratively perturbing input samples and checking the classifier's output until samples are misclassified. Other methods like LBFGS [19], and C&W [20] aimed to generate adversarial examples using different approximations to find the shortest distance. LBFGS and C&W methods are based on approximations to the shortest distance, and replacing the constraint with a penalty. These methods usually require multiple iterations and gradient calculations, which are time-consuming. In this work, we comprehensively compare our DDB calculation method with these methods to show the advantage of our method for the cooperative spectrum sensing scenario.

VI. Conclusion

In this paper, we presented a novel defense method for detecting malicious sensing values in cooperative spectrum sensing. Our approach leverages the distribution of the distance from a sensing value to the decision boundary of a classification algorithm, providing an effective means of detecting adversarial attacks. The experimental results demonstrate the advantages in terms of the efficiency and effectiveness of our proposed method over existing approaches for calculating the distance to the decision boundary. By effectively detecting adversarial attacks, our method can improve the overall performance of spectrum sensing systems.

Acknowledgement: The work at University of South Florida was supported in part by NSF Grants 2029875 and 2044516. The work at University of Miami was supported in part by NSF Grant 2029858.

REFERENCES

[1] P. J. Kolodzy, "Dynamic spectrum policies: promises and challenges," *CommLaw Conspectus*, 2004.

[2] R. I. Chiang, G. B. Rowe, and K. W. Sowerby, "A quantitative analysis of spectral occupancy measurements for cognitive radio," in *2007 IEEE 65th Vehicular Technology Conference-VTC2007-Spring*, 2007.

[3] E. C. Peh, Y.-C. Liang, Y. L. Guan, and Y. Zeng, "Cooperative spectrum sensing in cognitive radio networks with weighted decision fusion schemes," *IEEE Transactions on Wireless Communications*, 2010.

[4] S. Atapattu, C. Tellambura, and H. Jiang, "Energy detection based cooperative spectrum sensing in cognitive radio networks," *IEEE Transactions on wireless communications*, 2011.

[5] Z. Luo, S. Zhao, Z. Lu, J. Xu, and Y. Sagduyu, "When attackers meet AI: Learning-empowered attacks in cooperative spectrum sensing," *IEEE Transactions on Mobile Computing*, 2020.

[6] H. Li, Y. Gu, J. Chen, and Q. Pei, "Speed adjustment attack on cooperative sensing in cognitive vehicular networks," *IEEE Access*, 2019.

[7] J. Yi, C. Poellabauer, X. S. Hu, T. Chantem, and L. Zhang, "Dynamic channel reservations for wireless multihop communications," *ACM SIGMOBILE Mobile Computing and Communications Review*, 2010.

[8] C. Liu, J. Wang, X. Liu, and Y.-C. Liang, "Deep CM-CNN for spectrum sensing in cognitive radio," *IEEE Journal on Selected Areas in Communications*, 2019.

[9] J. Xie, C. Liu, Y.-C. Liang, and J. Fang, "Activity pattern aware spectrum sensing: A CNN-based deep learning approach," *IEEE Communications Letters*, 2019.

[10] H. Wang and Y.-D. Yao, "Primary user boundary detection in cognitive radio networks: Estimated secondary user locations and impact of malicious secondary users," *IEEE Transactions on Vehicular Technology*, 2018.

[11] S. Rajasegarar, C. Leckie, and M. Palaniswami, "Pattern based anomalous user detection in cognitive radio networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[12] M. Liu, H. Zhang, Z. Liu, and N. Zhao, "Attacking spectrum sensing with adversarial deep learning in cognitive radio-enabled internet of things," *IEEE Transactions on Reliability*, 2022.

[13] Y. Vorobeychik and M. Kantarcioglu, "Adversarial machine learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2018.

[14] S. Zheng, L. Ye, X. Wang, J. Chen, H. Zhou, C. Lou, Z. Zhao, and X. Yang, "Primary user adversarial attacks on deep learning-based spectrum sensing and the defense method," *China Communications*, 2021.

[15] D. Mickisch, F. Assion, F. Greßner, W. Günther, and M. Motta, "Understanding the decision boundary of deep neural networks: An empirical study," *arXiv preprint arXiv:2002.01810*, 2020.

[16] H. Karimi, T. Derr, and J. Tang, "Characterizing the decision boundary of deep neural networks," *arXiv preprint arXiv:1912.11460*, 2019.

[17] W. He, B. Li, and D. Song, "Decision boundary analysis of adversarial examples," in *International Conference on Learning Representations*, 2018.

[18] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[20] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*, 2017.

[21] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, "Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[22] F. J. Massey Jr, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American statistical Association*, 1951.

[23] A. G. Fragkiadakis, E. Z. Tragos, and I. G. Askoxylakis, "A survey on security threats and detection techniques in cognitive radio networks," *IEEE Communications Surveys & Tutorials*, 2012.

[24] R. Chen, J.-M. Park, and K. Bian, "Robust distributed spectrum sensing in cognitive radio networks," in *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*, 2008.

[25] N. Nguyen-Thanh and I. Koo, "A robust secure cooperative spectrum sensing scheme based on evidence theory and robust statistics in cognitive radio," *IEICE transactions on communications*, 2009.

[26] Y. Zhang, A. Li, J. Li, D. Han, T. Li, R. Zhang, and Y. Zhang, "Speckriging: GNN-based secure cooperative spectrum sensing," *IEEE Transactions on Wireless Communications*, 2022.

[27] Z. Li, Z. Xiao, B. Wang, B. Y. Zhao, and H. Zheng, "Scaling deep learning models for spectrum anomaly detection," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2019.

[28] S. Xu, Y. Shang, and H. Wang, "Double thresholds based cooperative spectrum sensing against untrusted secondary users in cognitive radio networks," in *VTC Spring 2009-IEEE 69th Vehicular Technology Conference*, 2009.

[29] H. Chen, M. Zhou, L. Xie, and J. Li, "Cooperative spectrum sensing with M-ary quantized data in cognitive radio networks under SSDF attacks," *IEEE Transactions on Wireless Communications*, 2017.

[30] Z. Luo, S. Zhao, R. Duan, Z. Lu, Y. E. Sagduyu, and J. Xu, "Low-cost influence-limiting defense against adversarial machine learning attacks in cooperative spectrum sensing," in *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*, pp. 55–60.

[31] B. Kim, Y. E. Sagduyu, K. Davaslioglu, T. Erpek, and S. Ulukus, "Channel-aware adversarial attacks against deep learning-based wireless signal classifiers," *IEEE Transactions on Wireless Communications*, 2021.

[32] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[33] K. Nar, O. Ocal, S. S. Sastry, and K. Ramchandran, "Cross-entropy loss and low-rank features have responsibility for adversarial examples," *arXiv preprint arXiv:1901.08360*, 2019.

[34] G. Elsayed, D. Krishnan, H. Mobahi, K. Regan, and S. Bengio, "Large margin deep networks for classification," *Advances in neural information processing systems*, 2018.

[35] D. Mishkin and J. Matas, "All you need is a good init," *arXiv preprint arXiv:1511.06422*, 2015.

[36] R. H. Coase, "The federal communications commission," *The Journal of Law and Economics*, 1959.

[37] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[38] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy *et al.*, "Technical report on the cleverhans v2. 1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2016.

[39] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "Ead: elastic-net attacks to deep neural networks via adversarial examples," in *Proceedings of the AAAI conference on artificial intelligence*, 2018.

[40] H. Tang, F. R. Yu, M. Huang, and Z. Li, "Distributed consensus-based security mechanisms in cognitive radio mobile ad hoc networks," *IET communications*, 2012.

[41] Q. Yan, M. Li, T. Jiang, W. Lou, and Y. T. Hou, "Vulnerability and protection for distributed consensus-based spectrum sensing in cognitive radio networks," in *2012 Proceedings IEEE INFOCOM*, 2012.

[42] H. A. B. Salameh, S. Almajali, M. Ayyash, and H. Elgala, "Spectrum assignment in cognitive radio networks for internet-of-things delay-sensitive applications under jamming attacks," *IEEE Internet of Things Journal*, 2018.

[43] J. Ren, Y. Zhang, Q. Ye, K. Yang, K. Zhang, and X. S. Shen, "Exploiting secure and energy-efficient collaborative spectrum sensing for cognitive radio sensor networks," *IEEE transactions on wireless communications*, 2016.

[44] D. Adesina, C.-C. Hsieh, Y. E. Sagduyu, and L. Qian, "Adversarial machine learning in wireless communications using RF data: A review," *IEEE Communications Surveys & Tutorials*, 2022.

[45] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," *arXiv preprint arXiv:2001.03994*, 2020.

[46] C. Finlay and A. M. Oberman, "Scaleable input gradient regularization for adversarial robustness," *arXiv preprint arXiv:1905.11468*, 2019.