future internet

MDPI

Article

PUE Attack Detection by Using DNN and Entropy in Cooperative Mobile Cognitive Radio Networks

Ernesto Cadena Muñoz [1,*], Gustavo Chica Pedraza [1], Rafael Cubillos-Sánchez [1], Alexander Aponte-Moreno [1] and Mónica Espinosa Buitrago [2]

[1] School of Telecommunications Engineering, Universidad Santo Tomás, Bogotá 110311, Colombia; gustavochica@usantotomas.edu.co (G.C.P.); rafaelcubillos@usantotomas.edu.co (R.C.-S.); jhonaponte@usantotomas.edu.co (A.A.-M.)

[2] School of Electronics Engineering, Universidad de Cundinamarca, Fusagasuga 252211, Colombia; mespinosab@ucundinamarca.edu.co

* Correspondence: ernestocadena@usantotomas.edu.co

Abstract: The primary user emulation (PUE) attack is one of the strongest attacks in mobile cognitive radio networks (MCRN) because the primary users (PU) and secondary users (SU) are unable to communicate if a malicious user (MU) is present. In the literature, some techniques are used to detect the attack. However, those techniques do not explore the cooperative detection of PUE attacks using deep neural networks (DNN) in one MCRN network and with experimental results on software-defined radio (SDR). In this paper, we design and implement a PUE attack in an MCRN, including a countermeasure based on the entropy of the signals, DNN, and cooperative spectrum sensing (CSS) to detect the attacks. A blacklist is included in the fusion center (FC) to record the data of the MU. The scenarios are simulated and implemented on the SDR testbed. Results show that this solution increases the probability of detection (PD) by 20% for lower signal noise ratio (SNR) values, allowing the detection of the *PUE attack* and recording the data for future reference by the attacker, sharing the data for all the SU.

Keywords: cognitive radio networks; cooperative spectrum sensing; deep learning; multiple PUE attack; primary user emulation

Citation: Muñoz, E.C.; Pedraza, G.C.; Cubillos-Sánchez, R.; Aponte-Moreno, A.; Buitrago, M.E. PUE Attack Detection by Using DNN and Entropy in Cooperative Mobile Cognitive Radio Networks. *Future Internet* **2023**, *15*, 202. https://doi.org/10.3390/fi15060202

Academic Editor: Nikos Fotiou

Received: 28 April 2023
Revised: 22 May 2023
Accepted: 26 May 2023
Published: 31 May 2023

1. Introduction

In the era of Industry 4.0, applications, data management, and data analysis are needed. Technologies such as the Internet of Things (IoT), augmented or virtual reality, big data, and artificial intelligence need wireless technology that efficiently improves throughput, security, and spectrum access. Entities such as the Federal Communication Commission (FCC) recognize that part of the assigned radio-electric spectrum is not being used; its utilization can be under 15%. Cognitive radio networks (CRN) are a possibility for increasing the use of this spectrum by allowing a SU to use the spectrum if a PU is not using it, which requires continuous monitoring of the spectrum's use in a specific frequency range. If a PU starts to use a frequency, the SU must move to another frequency if there is another spectral hole in the assigned space. The spectral hole is a frequency channel not used at a specific time and can be used for the PU or SU. For example, if there are four free frequency channels to assign, a SU can use one of them, but if a PU wants to use it, the SU changes to any free channel of the three that remain or waits for one of them to be released to use it [1].

One possible solution to spectrum scarcity is the implementation of a CRN. In mobile networks, for example, there are time slots where the spectrum frequencies are unused, partly because their assignment is fixed by the regulatory entities in a traditional spectrum management system, giving the operators flexibility in their usage. However, if we were to implement the CRN, it would be crucial to acknowledge the accompanying security

threats, including both traditional risks and those arising from the SU's random access to free channels for its own purposes [2].

The users in this environment are the PU and the SU, and then another user might appear, the MU, which uses radio signals to emulate the PU signal. The SU releases the occupied channel because they detect a PU, and the MU obtains access by cheating the SU. In the CRN environment, this process of searching the spectrum holes is called spectrum sensing and is one of the most affected processes in the network. However, all parts of the process are affected by this *PUE attack*. We must ensure that the SU uses empty spectrum bands without affecting the PU. To secure this, the CRN needs to check if the channel is occupied, coordinate with other users to access the spectrum, and release the channel if a PU starts transmitting its data [3].

One SU that detects the *PUE attack* is affected by signal characteristics such as shadowing and multipath fading. Then, we can improve the detection system by including the CSS, which has a better probability of detecting the attack using several SU in a centralized or distributed architecture [4]. In CSS, each SU identifies the radio environment and looks for a PU signal; this local sensing information is sent to the FC, which centralizes all the SU information. The single detection of each one takes a global decision based on an established rule. For this scenario, the SU can send a simple bit with the decision about the PU presence or the averaged energy to the FC; this is called a hard or soft combination rule, respectively [1].

Recently, artificial intelligence (AI), which includes machine and deep learning, has been used for PUE detection; it shows an increase in detection performance compared with conventional schemes. It is close to the optimal strategy for classifying the attack data learned from many sample data sets. These algorithms can be applied in wireless communication systems for classifying communication, and they are more adaptive than conventional CSS schemes. Convolutional neural networks (CNNs) have been used for SU spectrum sensing, and CSS schemes based on machine learning (ML) models, such as the support vector machine (SVM), have been studied in [5].

The DNN has been used in PUE attacks to decrease the probability that a MU affects the CRN system. The algorithm is trained with the data rate, distance, and power and compared with the K-nearest neighbor (KNN) classifier, making a classification process. The trained classifier detects the PUE with high performance; simulation results show better accuracy than conventional PUEA classification techniques, including the KNN [6].

In our previous work, we used entropy detection for spectrum sensing [7], and in [8], we explored the CSS for spectrum sensing but not for detecting the *PUE attack*. In [9] and [10], we use the SVM and KNN to detect PUE attacks with a PUE detector implemented in a single SU. In this paper, we propose a novel DNN algorithm that takes the entropy of the energy detector in a cooperative scenario with CSS, which gives the cognitive system a better sense of the entire network environment and results in lower SNR values. After a learning process, the DNN can detect the PUE in a MCRN, which includes the PUs, three SU, and a PUE attacker. The algorithms are implemented in an SDR device, and the attack is generated in an MCRN network with a frequency hopping protocol in GNU Radio. The DNN algorithm has not been used in the literature to detect PUE attacks in a MCRN environment with a CSS and with SDR implementation; we evaluated the results with other detection techniques in a realistic environment.

The paper is organized into six sections. Section 2 analyzes the previous work and contributions. In Section 3, the DNN and CSS detection schemes are illustrated. Section 4 focuses on the model and scenarios. Section 5 shows the experiment results and discussion, and Section 6 shows the conclusions.

2. Previous Work

The *PUE attack* is caused when the PUE attacker tries to mimic the honest PU signal and emulates it, corrupting the data transmitted to the FC. PUE attacks are classified as always on and smart. Always on does not know the status of the PU; it sends the PUE

signals in the PU frequency band. Smart PUEs know the PU signal status and attack the network depending on the PU's presence. The SUs report wrong decisions to the FC, leading to a bad global decision. The PUE uses a free frequency for its signal or causes interference, which is called selfish or malicious PUE, respectively [11].

For PUE detection, there have been some schemes; some of them are individual detection, and other works explore the CSS. In [7], we propose a method that uses the entropy calculation of the received signal for spectrum sensing; the results increase the performance for low SNR values, and it works in a simple SU. The basis of this detection is the spectrum sensing system. There are several problems, such as multipath fading, shadowing, and receiver uncertainty, and in an MCRN, there are also different modulation types, and it is difficult to distinguish between them. A possible solution for some of these problems is to share the information of all the users of the CRN; this means sharing the spectrum sensing data, the detected signal, frequency, and bands of use since these data increase detection performance, providing a solution that can be implemented for an MCRN [6]. We implemented this solution only for spectrum sensing and not for PUE detection [8].

The CSS model has been used to improve the probability of detection by including cooperation in the SUs, which share the particular sensing data and make a combined decision using an AND, OR, or MAJORITY rule, obtaining better results than the individual decisions of each CR and giving the MCRN a better knowledge of the radio environment. Energy-based detection is the faster and simpler method for spectrum sensing in the proposed CSS schemes. This method uses an energy threshold value to detect the presence of the PU/PUE in the radio environment, or with a test statistic and two thresholds, which obtain better results for sense. This is helpful to detect the signal, but it is not able to detect PUE attacks by itself [7].

In the CSS models, some authors have designed strategies to use the hard or soft fusion methods using MATLAB algorithms in additive white Gaussian noise (AWGN) or Rayleigh channels. The results show that the best method is a soft combination. Another platform used for implementing it on a device due to the costs and keeping high computational performance is the Raspberry Pi. Additionally, the SDR is another choice for CRN implementation [12].

The CSS model can be centralized using the FC; the cooperating SUs send their signal measurements or decisions about a sensing process to an FC or a base station BS. Then, the FC takes a general decision and transmits it to all the SUs. Another model uses a decentralized CSS, which is based on the sharing of data or decisions by the SUs to make a final decision. Still, it needs a radio frequency for the communication of each SU, requiring more radio resources. The authors use a centralized CSS to improve performance in deep fading environments with low SNR values. The PUE is a MU that reconfigures its air interfaces and transmits fake signals, leading the CSS system to make a wrong decision due to the SU report to the FC due to the openness of the lower layers [11].

Furthermore, apart from CSS, there are other techniques worth considering in the implementation of CRN. These techniques include the implementation of optimal joining techniques to detect PUEAs, detecting the MU, and bridging the gap between single equilibrium and social strategies through the imposition of a selected fee on SUs [13]. The time difference of arrival (TDOA) is another method used to find the PUE attacker position, but in an MCRN, this technique will not work because users are in motion. SUs in the CRN cooperate based on position estimation using the TDOA. Another author uses a proficient TDOA localization algorithm using the Differential Evolution (DE) method to detect the PUEA in CRNs, showing fast convergence of the detection [2].

Particle swarm optimization (PSO) has been used for PUE detection. The author in [14] uses a three-phase detection system: the state of the spectrum is calculated, the SVM algorithm is used to estimate if a frequency is occupied, and a PSO makes the selection of kernel and bias values. The technique is evaluated in MATLAB. In [15], the authors detect the PUE using the TDOA localization technique combined with a PSO algorithm to solve

the cost function based on the TDOA results. It changes the parameters such as inertia, weight, and acceleration, but if a PUE, a SU, or a PU is in motion, these techniques will not work because they are based on the fixed position of the users.

The AI has been used for the detection of PUE. In [16], the authors introduce a classification method called Online Adaptive Memory-Based Genetically Optimized Artificial Neural Network (OAM-GANN), based on online learning, that uses the network parameters to identify the PUE. It optimally tunes the hyperparameters of the developed DNN. It improves the security of the CRN. It is evaluated by several metrics, such as error rate and detection probability [16].

Other solutions are based on ML algorithms, such as the Knearest neighbor classifier (KNN) algorithm, which can identify the *PUE attack* and group the malicious nodes together [17]. In another work, we compare KNN, random forest, and SVM, showing that for this problem, the best results are achieved with the SVM in an MCRN with a single SU [9]. Extreme machine learning (EML) and time–distance with signal strength evaluation (TDSE) methods have been used to detect or prevent PUE attacks. The TDSE implementation allows for identifying the malicious PUE attacker, and the EML compares it with the MU, but its computational requirements are high for a mobile network [18].

Another author uses the logistic regression concept to estimate the maximum likelihood of detecting the attack. The algorithm is divided into two parts: the training and testing processes, based on a dataset generated with an active *PUE attack*. Results show a high PD of about 99.5% with a PFA of less than 0.6% [5].

In another work, the authors implemented a classification problem and solved it with an artificial neural network (ANN). ANN increases its performance by using the immune plasma optimization (IPO) method. The results of this work show that the algorithm using the ANN has better results in terms of accuracy and other network variables. This method has an accuracy rate improvement of 32% and a 16% energy savings compared to the existing methods [19].

Other authors have proposed the DNN algorithm by combining energy detection, cyclostationary calculation, and the DNN algorithm. The PU/PUE in the frequency band is localized by using energy detection. Then, the cyclostationary technique identifies the features of the signal, and the results are used as input for the neural network to detect the *PUE attack*. Other work shows that radio-frequency fingerprinting can be used to detect the *PUE attack*. In the training step, the transmitter profiles are elaborated for each one, and a classifier with three layers of the ANN is created. The final part of the process is to compare the user signal to the established profiles to determine the PUE presence [5].

Most of the related previous work uses energy detection as the base for signal detection; we propose entropy detection for the *PUE attack* because it is able to work at lower SNR values with a high probability of detection. The CSS has been worked on in simulations, but there have been no results with a realistic SDR environment as we implemented it in our work. Finally, in the literature, there are no implementations of the DNN algorithms to detect the *PUE attack*. In this work, we include the algorithms of the DNN in the SU and test them in an SDR environment with mobile phones.

3. Deep Neural Network and Entropy to Multiple PUE Detection in Cooperative MCRN

The following subsections describe the CSS scheme, the entropy detector, the DNN algorithm, and the general proposal for *PUE attack* detection.

3.1. Cooperative Spectrum Sensing Model

The designed model has a malicious *PUE attack*, some SUs and PUs in the network, and an FC for the centralized CSS. There is one radio frequency channel of the PU detected by the SU, another one of the PUE attacks detected by the SU, and one common channel where the SUs communicate with the FC; these signals can be seen in Figure 1.

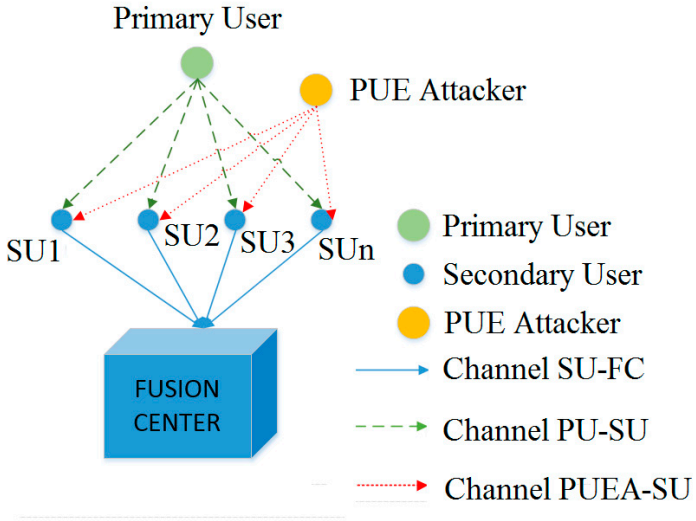

Figure 1. System model for cooperative spectrum sensing.

For cooperative spectrum sensing, each SU detects the *PUE attack* based on the entropy and DNN techniques described in the following subsections and informs the FC of the results. The FC takes an OR decision that helps the system detect any attack within the network coverage and the assigned frequencies. The FC manages the frequencies for detection and the state of each frequency in the MCRN.

3.2. Detector Based on Entropy

An entropy-based detector is used for the detection; the energy detection results are the inputs for the system. The MCRN has n SUs, which transmit the individual results to the FC with the calculated entropy in one frequency channel. These entropy results are the input for the DNN learning algorithm, which detects if there is a *PUE attack* in the frequency and communicates it to the FC, which decides for everyone under an established rule and in the presence of a *PUE attack*. If a *PUE attack* is detected, SU continues using the frequency, but if there is a *PU*, it must release the channel and use another one, and if there are no more available frequency channels in the range, service cannot be provided.

Energy detection is used for the spectrum sensing scheme in the SUs, but it is combined with the entropy detector for better results, as we worked out in [7]. N is the number of samples; then the test statistic is given in Equation (1) [20], where $p(n)$ is the power of the signal received at the SU:

$$T(p) = \frac{1}{N}\sum_{n=1}^{N} |\, p(n)|^2 \tag{1}$$

In the traditional energy detector, a hypothesis test is used, a threshold is defined, and it is analyzed if it is above or below the threshold to detect if a *PU*/*PUE* is present. Our proposal uses the averaged energy as an input for the entropy detector. The *PU*/*PUE* signal is described in Equation (2) [7].

$$p(t) = \begin{cases} n(t) & noise \\ h(t) * o(t) + n(t) & PU/PUE \end{cases} \tag{2}$$

where $n(t)$ is the noise and $h(t)$ is the channel impulse response of the system, which is the response of the communication channel when an impulse is applied to the system. It can be seen as the correlation of the received signal against the transmitted signal, where $o(t)$ is the measured data from a *PU*/*PUE* signal and $p(t)$ is the received signal. To detect a PU or *PUE attack* signal, it is necessary to protect the authentic PU by changing the frequency channel to another one. In contrast, the *PUE attack* is detected or discarded. For this purpose, a binary hypothesis test is used: H_0 is noise, and H_1 indicates a *PU*/*PUE* signal

and noise; this is defined in Equation (3) [7]. The entropy is calculated using a discrete Fourier transform (DFT), obtaining Equation (3) [7].

$$\overline{P}(k) = \begin{cases} \overline{J}(k) & H_0 \\ \overline{L}(k) + \overline{J}(k) & H_1 \end{cases} \tag{3}$$

where $\overline{P}(k)$ is the complex spectrum of the received signal, $\overline{L}(k)$ is the *PU/PUE* signal, $\overline{J}(k)$ is the noise, and the size for the DFT is $k = N$, where N is the sample size. H_0 implies the presence of a *PU*/*PUE* signal, and H_1 its absence [21].

The results of the entropy calculation on each SU of the CSS using Equation (4) are used as an input for the DNN algorithm in the learning and detecting processes; the binary results of the detection are sent to the FC as we worked it out in [8]. For a given number of bins X, using the histogram method, the statistic test is calculated in Equation (4).

$$T(P) = \frac{1}{1-\alpha} \times log\left(\sum_{i=1}^{X} \frac{k_i}{N}^{\alpha}\right) \begin{Bmatrix} \leq \lambda : H_1 \\ > \lambda : H_0 \end{Bmatrix} \tag{4}$$

where N is the number of samples and λ is the entropy threshold calculated with the entropy H_X for X bins and described in Equation (5). The probability of a false alarm is P_{FA} and σ_e is the standard deviation [8].

$$\lambda = H_X + Q^{-1}(1 - P_{FA}) \times \sigma_e \tag{5}$$

The cooperative results are measured for the probability of detection (*Qd*) and the probability of false alarm (*Qfa*) measured at FC and are calculated by Equations (6) and (7) [20].

$$Q_d = 1 - \prod_{i=1}^{Z} (1 - P_{d,i}) \tag{6}$$

$$Q_{fa} = 1 - \prod_{i=1}^{Z} (1 - P_{fa,i}) \tag{7}$$

where Z is the number of SUs in the CSS. In the experiments, measurements are taken based on the results of putting the devices in different positions and distances.

3.3. Deep Learning Techniques for the Decision of PUE Presence

Deep learning is one of the most effective tools for making accurate forecasts using large and complex data sets. Deep learning includes technologies with multiple data types, including numbers, text, audio, images, video, or combinations [22]. We propose to use one of the deep learning techniques to help the SU detect the *PUE attack*; it must be implemented on each SU to improve the decision in a CSS scheme. In this section, an explanation of the context and deep learning techniques is given, and after this, we propose a deep learning scheme for detecting the *PUE attack*.

The central concept of deep learning is the use of neural networks, and even though their applications are widely used today, this technology is familiar, having emerged in the 1950s. However, its development was limited after the 1970s due to a lack of hardware and algorithms necessary to achieve the great expectations generated around neural networks. AI has been used in many fields, such as robots, mobile communications, autonomous cars, smart cities, financial analysis, and engineering problems such as decision support [23]. In 2012, there was a significant advance in the field of artificial intelligence, which allowed the creation of neural networks capable of identifying images with greater precision than humans. Later AI examples include Deep Blue, a chess-playing system that defeated Gary Kasparov, or Google's AlphaGo, which defeated the world's No. 1 ranked player Jie Ke in a Go match. In 2022, OpenAI launched Chat Generative Pre-trained Transformer (ChatGPT), a next-generation model that generates human-like responses based on text input [23].

In this case, we will use the neural networks for decision-making in an MCRN under a *PUE attack*.

To decide which neural network is needed, the ANN concept is analyzed. The fundamental components of the nervous system in biological systems are neurons; the complexity of this system derives from the millions of neuron connections when they communicate with each other through axons and dendrites. When a living organism learns, connections between neurons are established through a process called synapses [22]. Artificial neural networks are inspired by biological behavior through artificial neurons that communicate with each other and can learn from data. A neuron performs a linear regression from a set of weighted inputs and a bias [23].

There are different types of neural networks, each with its own architecture [24].

- Deep or fully connected networks (DNN): These are networks in which each neuron of a layer is connected with all the neurons of the subsequent layer; they have been used mainly in regression problems and the classification of supervised learning [25].
- Convolutional Networks (CNN): These networks, in which a set of filters is implemented in each layer, are very efficient in image processing.
- Recurrent Networks (RNN): In these models, the output depends on the current input and the information processed in the last moments. This characteristic, called short- and long-term memory, allows them to work with data series such as text, audio, or video. They are often used in natural language processing tasks, voice recognition, and time series analysis, among other applications [24].
- Generative Adversarial Networks (GAN): They are a system comprising two neural networks: generative and discriminative. The generative network creates new data from the training data set. In contrast, the discriminative network judges whether the data it receives is training data or data created by the generative network. As this competition process between the two networks progresses, each performs its task more efficiently. Thanks to this architecture, GANs have proven capable of generating high-quality images, music, and videos [26].

Deep neural networks are an effective tool for classification or regression tasks. These networks can identify complex patterns that relate input data to output values. In deep neural networks, three types of layers can be distinguished: the input layer, which has as many neurons as variables present in the training data; the deep hidden layer or layers; and the output layer, as shown in Figure 2 [27].

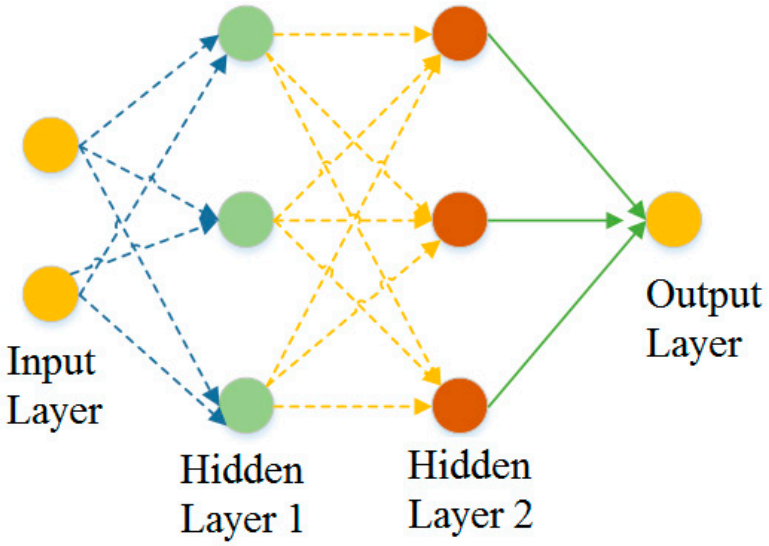

Figure 2. Deep artificial neural network.

In this paper, we propose to use a DNN; the input layers are the SNR and entropy of the measured signals, and with this data, each SU can detect the *PUE attack* using supervised learning and an adequately labeled dataset. This data set is divided into training and test or validation data. The network training stage follows an iterative process that includes several phases:

Initialization of parameters: Initially, the weights and biases of each neuron in the network are adjusted. These values can be chosen randomly or they can be assigned according to a probability distribution.

Forward propagation: In forward propagation of an L-layer neural network, a Z-weighted sum of the X inputs and the b biases of each neuron in the layer is performed, where the weights w represent the contribution of each input in the layer in the weighted sum, as can be seen in Equation (8). In this case, the inputs are the SNR and the entropy of the averaged energy of the received signal.

$$Z^L = w^L X + b^L \tag{8}$$

Subsequently, a nonlinear activation function a is applied to the weighted sum Z to obtain the response $\hat{y}$, as seen in Equation (9).

$$\hat{y} = a(Z^L) \tag{9}$$

Error calculation: The *error* is obtained by calculating the cost function C, which measures the discrepancy between the predicted output $\hat{y}$, and the desired output or label y. The cost function C can be the root mean square *error* (RMSE) in cases of regression or cross-entropy in classification problems; the *error* can be seen in Equation (10) [28].

$$Error = y - \hat{y} = C(a(Z^{[L]})) \tag{10}$$

Backpropagation: After the *error* calculation, the backpropagation phase begins, in which the weights and biases of each neuron are adjusted to achieve the desired output. The cost function is optimized using algorithms, such as gradient descent, to minimize the *error*. In this way, it seeks to achieve a solution that allows a better network generalization to make accurate predictions on new data, reducing the *error* in the last layer, as seen in Equation (11) [29].

$$\delta^L = \frac{\partial C}{\partial a^L} \frac{\partial a^L}{\partial z^L} \tag{11}$$

The backpropagation for the *error* can be seen in Equation (12).

$$\delta^{L-1} = W^L \delta^L \frac{\partial a^{L-1}}{\partial z^{L-1}} \tag{12}$$

Then, the derivatives of the biases are calculated in Equation (13).

$$\frac{\partial C}{\partial b^{L-1}} = \delta^{L-1} \tag{13}$$

Finally, the derivatives of the weights are also calculated in Equation (14).

$$\frac{\partial C}{\partial w^{L-1}} = \delta^{L-1} a^{L-2} \tag{14}$$

Optimizing the weights and biases of the neural network based on their derivatives, known as backpropagation, is carried out by the backpropagation of the error from the output layer to the previous layers. This process is repeated with all the training data in one epoch and is executed multiple times until the network can capture the patterns in the data. Then, the validation of the network is carried out using the test data to measure its

efficiency. If the performance is unsatisfactory, the hyperparameters are adjusted, and the entire procedure is repeated [29].

After all the DNN processes are achieved (training, testing, and validation), each SU can detect the *PUE attack* by taking some samples, as will be explained in the next section.

3.4. General Proposal for PUE Attack Detection

Our proposal starts with the entropy calculation of the energy detector results, which achieves better results for low SNR values and energy by itself on each SU. After that, a DNN is used and divided into two parts. In the first one, the PUE attacker is placed at different points of a mesh around the SU of the network; the collected data is used for the learning process of the DNN algorithm, which will be explained in the following subsection. Each SU has the results of the learning stage in its programming. In the second part, the PUE is positioned anywhere, and the decision results of each SU are sent to the FC, which makes the global decision if there is a *PUE attack* presence based on an OR rule. There are other rules, such as the AND or MAJORITY rule, but depending on the position of the SU and the PUE attacker, there is a high probability that only one SU will be able to detect the attack. The rule that allows better detection is the OR rule; if any of the SU detects the *PUE attack*, it will be transmitted to the entire network. An example will be shown in Section 5.2. The global decision is sent in a broadcast message to all the SU in the CSS scheme. The general model for the proposed PUE detection can be seen in Figure 3.

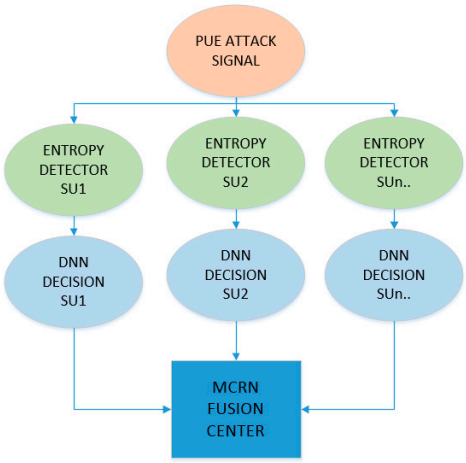

Figure 3. Proposed CSS using entropy and DNN for multiple PUE detection.

The model consists of a malicious *PUE attack* generated in the MCRN environment and some SUs that use energy and entropy methods to detect it, distributed on the MCRN. Depending on the position of a SU, it can detect or not the *PUE attack*; that is why the CSS needs to have a vision of the MCRN within its coverage limits. The malicious *PUE attack* does not send signals in uplink frequency (UL), only in downlink (DL) in the MCRN.

The first stage of the detection system is to analyze if a PU, a SU, or a MU is detected in the radio environment to protect the PU. However, to detect if there is a PU or a MU, the UL and DL signals are estimated. If there is a PU phone call, the two signals must be active, but if there is a MU, only a DL/UL signal will be detected. For the test, we use the absolute radio frequency channel number (ARFCN) and record the combination of UL/DL frequency pairs available for the phone call. If a DL signal is detected, the system detects the UL signal. If only the DL is detected by the system, it will be marked as a MU.

4. Experiments

A testbed based on SDR is used for the experiments; the Ettus N210 device is the SDR selected. It has been implemented for similar purposes and can work in the MCRN frequencies; it can be configured to work with Linux or Windows, and it has the libraries or

drivers to achieve several communication scenarios [30]. Each SU uses an RTL-SDR as a spectrum analyzer [12]. The N210 device generates the *PUE attack*; another SDR is used for the FC and each SU. The MCRN consists of some SU trying to make a phone call; this is achieved by using the OpenBTS software and GNURadio [31,32], as we work in [10]. The testbed for the experiments is shown in Figure 4.

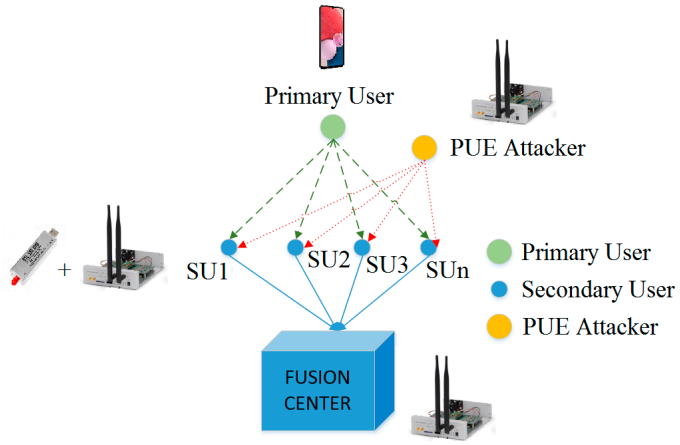

Figure 4. Testbed for experiments based on SDR, GNURadio, and OpenBTS.

The devices were positioned in a grid, and energy, entropy, and the DNN were calculated at that time and positioned with one PUE attacker to be able to measure the practical threshold. Then, the PUE attacker was moved to every place in that grid. That information is recorded and is the input for DNN algorithms. Once the measurements are taken and the algorithm learns how the attack works, the attacker takes a random position, and the SUs measure the results. With the energy and entropy results measured in each SU, the DNN predicts the *PUE attack* presence; this value is binary and transmitted to the FC. Notice that if any SU detects a signal, it changes its frequency immediately to prevent interference with a genuine PU while it detects the PUE's presence. The FC takes a global decision and transmits a broadcast message to any SU in the network; the whole system learns and records the attackers' information, creating a database with a blacklist of the attackers' data. For example, the grid of the device's position is shown in Figure 5.

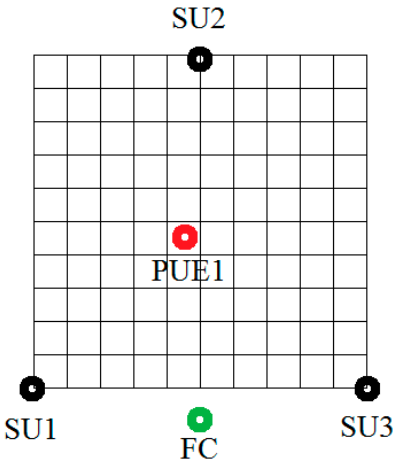

Figure 5. Example of the grid of the position of the device in the experiment.

If the PUE is out of the grid, it can detect it or not, depending on the SU position. The system needs at least one SU to detect it; if not, the PUE attacker is out of the system's coverage and will not be detected.

There are two parts to the process. The first part is where the algorithms learn how the attack is achieved and the threshold is measured. In this part, some samples are taken at each position, and with one PUE attacker, the frequency changes to detect some frequencies in the range. This learning process can be seen in Figure 6.

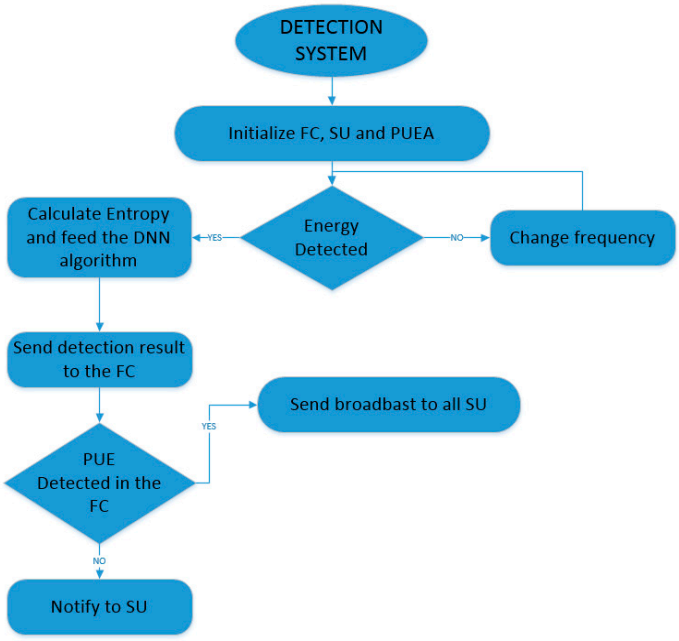

Figure 6. The learning system of the *PUE attack* detection flowchart.

The second part is the detection once the system has learned how to detect the *PUE attack*. This flowchart shows the steps that each SU takes. The *PUE attack* starts with a random position; this process can be seen in Figure 7.

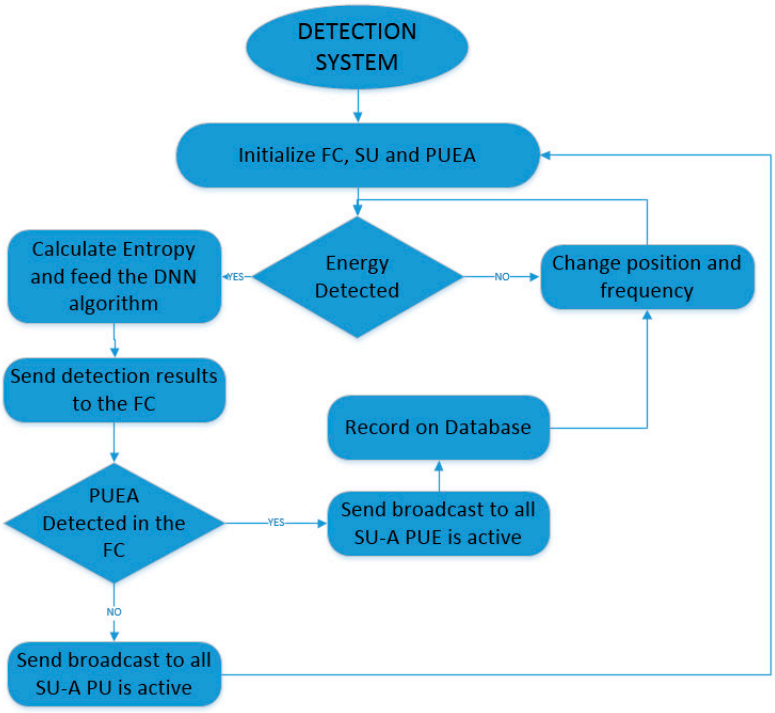

Figure 7. Detection system flowchart.

5. Results

The results are analyzed in this section, and a discussion about them is made.

5.1. Entropy Detection of PUE Attack

The first part of the model is the energy detector, which is implemented in GNURadio. The 100 positions of the grid are defined to measure the energy of one *PUE attack* when it is active or not. As shown in Figure 8, there is a difference in the power when it is in the two states.

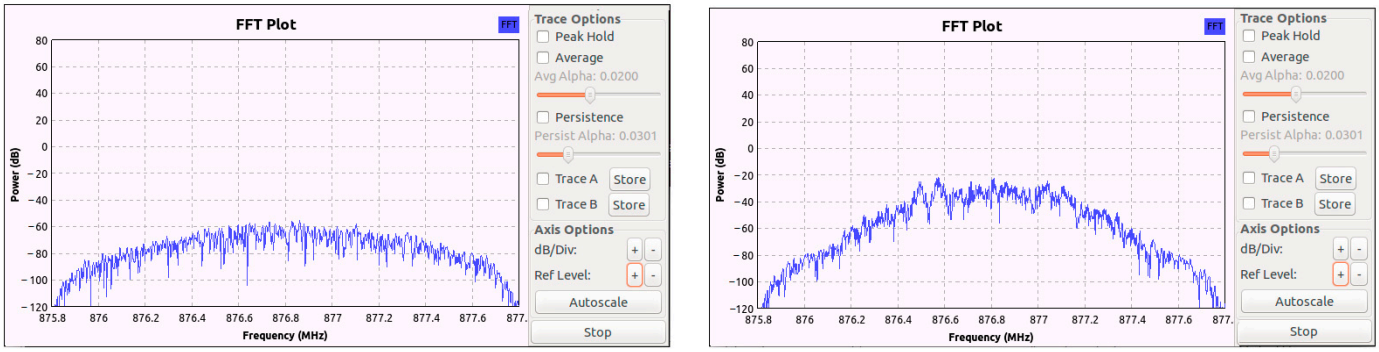

Figure 8. Example of Power Received in GNURadio.

The energy is detected by averaging 100 samples of the measured energy each time. Results are taken when the *PUE attack* is in a fixed position. In Figure 9, an example of the received power is shown when there is a PUE active or not in two different positions near and far from a SU.

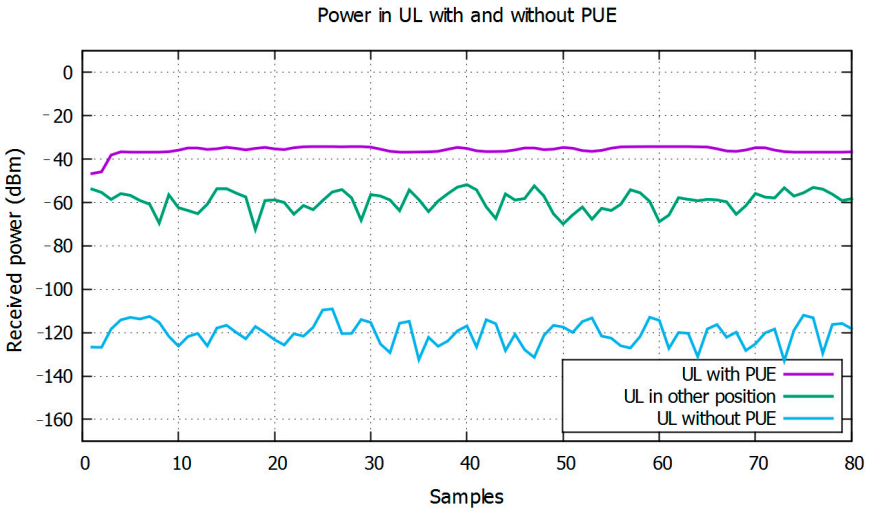

Figure 9. Power in UL when there is an active/inactive *PUE attack*.

The entropy is calculated with 100 samples of the energy. Figure 10 shows an example of the measured entropy when there is a PUE active or not close to the SU.

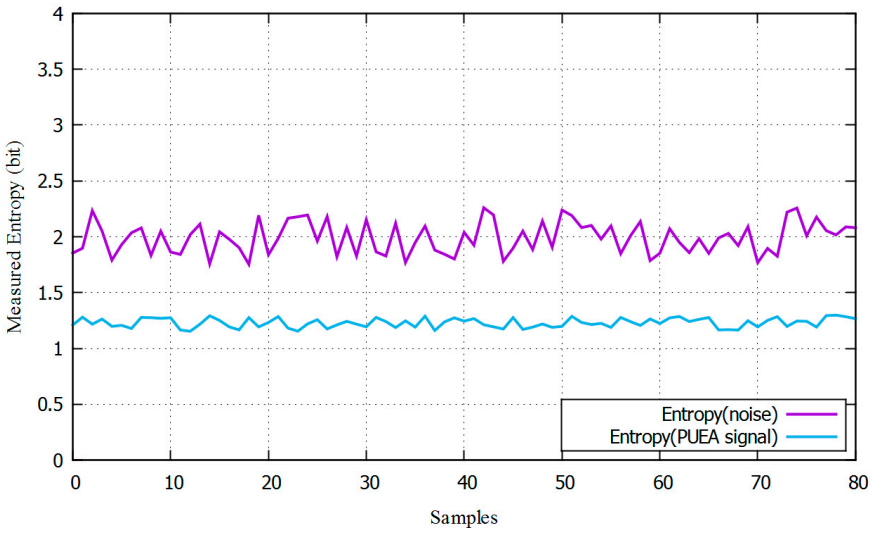

Figure 10. Measured entropy example with *PUE attack* signal and noise.

Now that the entropy values have been calculated, we can run the learning algorithm for the DNN. According to Table 1, the SNR is estimated for GNURadio.

Table 1. Parameters for the Learning Process.

Parameter	Value
Samples	20,000 samples
Averaged Values	100 samples for each point of the grid with an on/off value
Noise Signal	AWGN
Service	Phone Call-PUEA
Frequency	831.8 MHz
Confidence Level	95%
Margin of error	5%
Users	3 SU, 1 FC, 1 PUE

For the experiments, we use GSM-850 MHz, which is assigned to an operator in the laboratory's coverage zone. A site survey was made, and we found that ARFCN 166 is not used. This corresponds to a DL frequency of 876.8 MHz and a UL frequency of 831.8 MHz. However, during all the experiments, the MCRN was configured to release the channel if a *PU* was detected.

5.2. DNN Algorithm Results

Once the entropy and SNR results are taken, the DNN algorithm is trained with this data. For this purpose, the scikit-learn and Keras libraries of Python are used.

A classifier was built using a DNN (fully connected) in a sequential model implemented in Keras. Two deep layers were simulated with 16 and 32 neurons, and using the ReLu activation function, the output layer contains two neurons and a Softmax activation function. The final model had 642 parameters, optimized through a training process that covered 50 epochs and a batch size of 32, as shown in Figure 11. The input is the SNR and calculated entropy.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 16)                32

 dense_1 (Dense)             (None, 32)                544

 dense_2 (Dense)             (None, 2)                 66

=================================================================
Total params: 642
Trainable params: 642
Non-trainable params: 0
_________________________________________________________________
```

Figure 11. Parameters of DNN algorithm.

The algorithm is tested with 20% of the dataset when the training is performed. In this case, 20,000 samples are taken, so 4000 data points are used for the test, as shown in Figure 12 Values are measured for each SNR value from −25 dB to 0 dB. For example, for −12 dB, the algorithm's probability of detection and accuracy are 99%; the blue boxes are the right detection values.

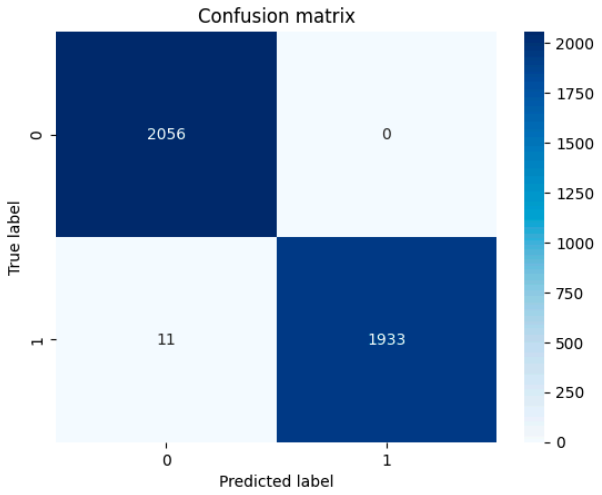

Figure 12. Confusion matrix for SNR = −12 dB.

After the training algorithm is executed, the results are shared and recorded on each SU in the system. Then, the second part of the DNN is executed with each value of entropy; the algorithm predicts if there is an active PUE, and depending on the estimated SNR, a probability of detection is calculated.

In the SDR experiments, the PUE position is random and fixed, the SNR is estimated, 100 entropy values are averaged on each sample, and then each value is processed by the SUs. A CSS with an OR logic is implemented; if any of the three SUs detect the PUE presence, the FC broadcasts a message of the PUE presence to any SU in the network. With the obtained data, the device records it on the blacklist of PUE attackers.

The receiver operating characteristic (ROC) curves are similar to the practical results; the experiments are conducted in laboratory conditions. An example of the implemented algorithm can be seen in Figure 13. The PUE is active, and the FC indicates that a *PUE attack* is present when any of the SUs detect it and send a "1" binary value. It can be seen that one SU detects the attack in some samples, which is why an OR rule is used.

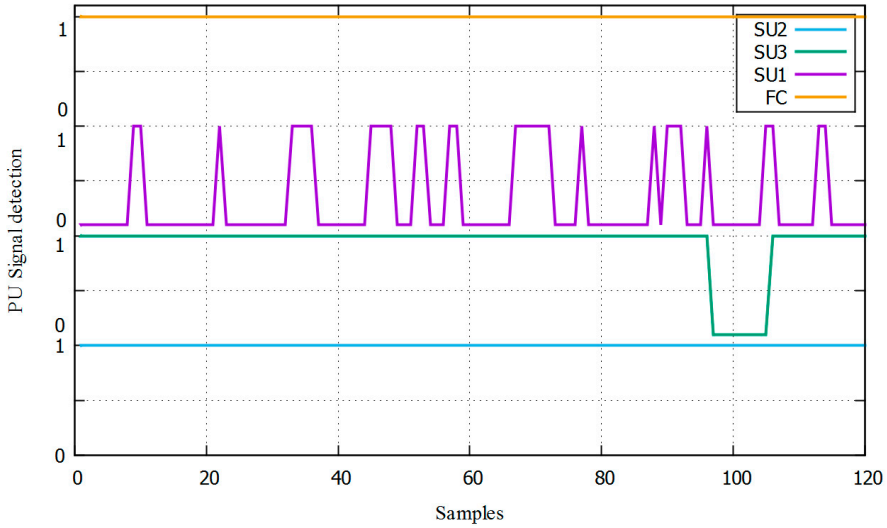

Figure 13. Example of the FC-or rule.

The probability of detection of the PUE is estimated as the number of positive detections when the PUE is active. In the experimental SDR testbed, the measurements are taken with steps of 1 dB each time, as seen in Figure 14 The results are compared with our previous results on entropy and the SVM *PUE attack* detection system [9].

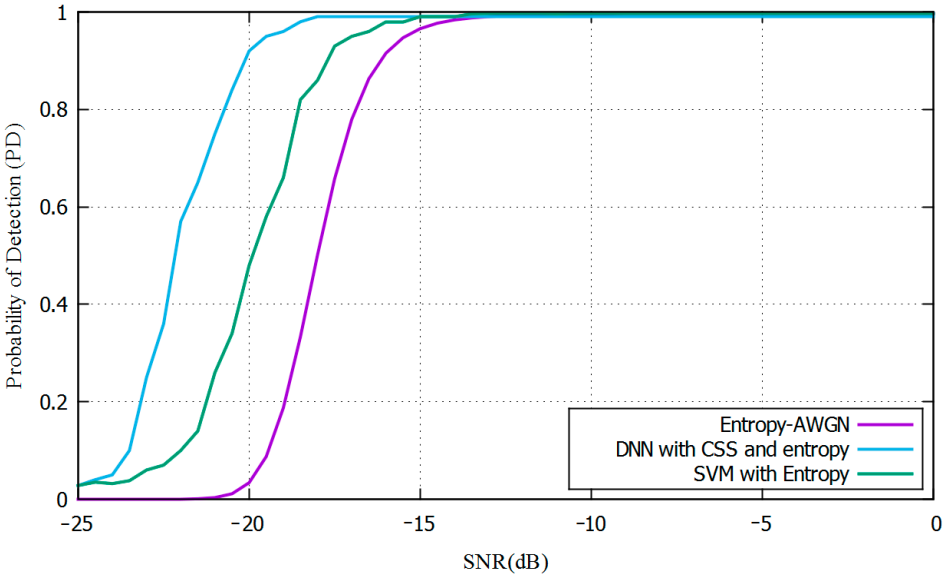

Figure 14. Probability of detection in comparison with SVM and entropy.

5.3. Discussion

The designed algorithms are an efficient combination of energy, entropy, DNN, and CSS models, which allow the detection system to obtain a PD of above 90% at −21 dB of SNR in an actual SDR environment. Compared to our previous works, such as [9], our proposal demonstrates improved outcomes regarding noise, SNR, and PD while maintaining a fast method to release the channel and prevent interference with the PU.

In the implemented model, there is no need to know previous information such as the modulation schemes, the threshold, or some earlier data similar to other solutions. The DNN is a high-speed method that, after a learning process, can detect PUE attacks. In combination with the CSS schemes, it increases the probability of detection by using the SUs as sensors and transmitting the individual detections to the FC. The use of entropy allows us to work with lower SNR values.

As a future work, the attack and detection methods could be implemented in a 5G scenario using, for example, a 5G testbed, as mentioned in [33]. There can be experiments with some mobile technologies, CRN, attacks, and detection methods that can be compared with the results of our proposal. Another detection method could be to use the Boltzmann concept in a game strategy, as demonstrated in [34].

6. Conclusions

The experimental results obtained in the SDR environment show that a cognitive protocol can be implemented to protect the primary user in MCRN by combining GNURadio, Python, and the DNN libraries. The CSS is used to detect the *PUE attack*, and according to the results, it works better with low SNR values. In combination with an entropy detector, it shows a higher probability of detection than all the systems, including the SVM. The implemented PUE detection solution increases the PD by about 10% for an SNR value of −18 dB and more than 20% at −21 dB, using the three SUs as detectors in the CSS scheme. If more than 90% PD is needed, this can be achieved with the DNN algorithm for an SNR value of −21 dB or higher.

The learning process takes several minutes to measure the variables at several distances and SNR values. Still, once it is performed, the detection system can be conducted significantly faster, which prevents the interference of the PU, which is one of the objectives of the MCRN. The system can release the channel in milliseconds, using the energy detector as the first step. At the same time, it is analyzed to determine if there is a PUE present in the radio environment. The use of the DNN to detect the PUE works better than other methods found in the literature [9].

Using the CSS gives us more coverage in the MCRN and allows us to obtain higher probability values for detection. Thanks to the DNN algorithm, high detection results are achieved even with slow SNR values. These results are similar to the values found in the literature. The system successfully detects the *PUE attack* signals individually and in a CSS scheme. In this case, the OR rule is better to have a better knowledge of the radio environment, separating the *PUE attack* signal and the MU efficiently because the FC has more data from the attacks detected on each SU, avoiding PUE attacks and errors in the network. A blacklist helps the system detect a previous PUE attacker without all the processes.

Author Contributions: The methodology, algorithms, and testbed proposed in this paper have been conceived by E.C.M.; the experiment scenarios by E.C.M., G.C.P. and R.C.-S.; testing by E.C.M., G.C.P., R.C.-S., A.A.-M. and M.E.B.; results and conclusion, review, and edition by E.C.M. and R.C.-S. and A.A.-M. All authors participated in the discussion and proofreading work. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Universidad Santo Tomas, Proyecto 2253501-AI, 2022.

Data Availability Statement: Not applicable.

Acknowledgments: We express our gratitude to Universidad Santo Tomas for the founding of this project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Khan, M.S.; Faisal, M.; Kim, S.M.; Ahmed, S.; St-Hilaire, M.; Kim, J. A correlation-based sensing scheme for outlier detection in cognitive radio networks. *Appl. Sci.* **2021**, *11*, 2362. [CrossRef]
2. Batool, R.; Bibi, N.; Muhammad, N.; Alhazmi, S. Detection of Primary User Emulation Attack Using the Differential Evolution Algorithm in Cognitive Radio Networks. *Appl. Sci.* **2022**, *13*, 571. [CrossRef]
3. Furqan, H.M.; Aygül, M.A.; Nazzal, M.; Arslan, H. Primary user emulation and jamming attack detection in cognitive radio via sparse coding. *EURASIP J. Wirel. Commun. Netw.* **2020**, *2020*, 141. [CrossRef]
4. Balogun, V.; Sarumi, O.A. A Cooperative Spectrum Sensing Architecture and Algorithm for Cloud-and Big Data-based Cognitive Radio Networks. In Proceedings of the 2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), London, ON, Canada, 30 August–2 September 2020; pp. 1–5.

5. Bliss Consultants, B. Detecting the Primary User Emulation Attack Using the Logistic Regression and MLE. 2018. Available online: https://ukdiss.com/examples/primary-user-emulation-attack.php?vref=1 (accessed on 26 May 2023).
6. Inamdar, M.A.; Kumaraswamy, H. Accurate primary user emulation attack (PUEA) detection in cognitive radio network using KNN and ANN classifier. In Proceedings of the 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 15–17 June 2020; pp. 490–495.
7. Cadena Muñoz, E.; Pedraza Martínez, L.F.; Hernandez, C.A. Rényi Entropy-Based Spectrum Sensing in Mobile Cognitive Radio Networks Using Software Defined Radio. *Entropy* **2020**, *22*, 626. [CrossRef]
8. Ernesto, C.M.; Martínez, J.A.R.; Martínez, L.F.P.; Parra, I.P.P. Cooperative Spectrum Sensing with Entropy for Mobile Cognitive Radio Networks. In Proceedings of the 2020 IEEE ANDESCON, Quito, Ecuador, 13–16 October 2020; pp. 1–5.
9. Muñoz, E.C.; Pedraza, L.F.; Hernández, C.A. Machine Learning Techniques Based on Primary User Emulation Detection in Mobile Cognitive Radio Networks. *Sensors* **2022**, *22*, 4659. [CrossRef]
10. Cadena Muñoz, E.; Pedraza Martínez, L.F.; Ortiz Triviño, J.E. Detection of Malicious Primary User Emulation Based on a Support Vector Machine for a Mobile Cognitive Radio Network Using Software-Defined Radio. *Electronics* **2020**, *9*, 1282. [CrossRef]
11. Shrivastava, S.; Rajesh, A.; Bora, P.K.; Chen, B.; Dai, M.; Lin, X.; Wang, H. A survey on security issues in cognitive radio based cooperative sensing. *IET Commun.* **2021**, *15*, 875–905. [CrossRef]
12. Villalonga, D.A.U.; Cotrina, E.G.; Salgado, A.A.V.; Gómez, J.T.; García, D.L. Cooperative Spectrum Sensing Application Using RTL-Dongle Technology. *Rev. Telemática* **2019**, *18*, 139–150.
13. Li, K.; Wang, J. Optimal Joining Strategies in Cognitive Radio Networks Under Primary User Emulation Attacks. *IEEE Access* **2019**, *7*, 183812–183822. [CrossRef]
14. Rajagopala, M.; Lingareddy, S. Spectrum occupancy-based PUEA detection using SVM-PSO in cognitive networks. *Int. J. Commun. Netw. Distrib. Syst.* **2021**, *26*, 30–49. [CrossRef]
15. Ghanem, W.R.; Mohamed, R.E.; Shokair, M.; Dessouky, M.I. Particle swarm optimization approaches for primary user emulation attack detection and localization in cognitive radio networks. *arXiv* **2019**, arXiv190201944.
16. Robert, V.N.J.; Vidya, K. OAM-GANN: Online Adaptive Memory Based Genetically Optimized Artificial Neural Network for PUEA Detection in CRN Applications, 12 August 2022, PREPRINT (Version 1). Available online: https://doi.org/10.21203/rs.3.rs-1952113/v1 (accessed on 26 May 2023).
17. Camana, M.R.; Garcia, C.E.; Koo, I.; Shakhov, V. Machine Learning Based Primary User Emulation Attack Detection. In Proceedings of the 2022 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Sofia, Bulgaria, 6–9 June 2022; pp. 244–248.
18. Sureka, N.; Gunaseelan, K. Investigations on detection and prevention of primary user emulation attack in cognitive radio networks using extreme machine learning algorithm. *J. Ambient Intell. Humaniz. Comput.* **2021**, 1–10. [CrossRef]
19. Ajay, V.; Nesasudha, M. Detection of Attackers in Cognitive Radio Network Using Optimized Neural Networks. *Intell. Autom. Soft Comput.* **2022**, *34*, 193–204. [CrossRef]
20. Charan, C. Double Threshold Based Cooperative Spectrum Sensing with Consideration of History of Sensing Nodes in Cognitive Radio Networks. In Proceedings of the 2018 2nd International Conference on Power, Energy and Environment: Towards Smart Technology (ICEPE), Shillong, Inida, 1–2 June 2018; pp. 1–9.
21. So, J. Entropy-based Spectrum Sensing for Cognitive Radio Networks in the Presence of an Unauthorized Signal. *KSII Trans. Internet Inf. Syst.* **2015**, *9*, 20–33.
22. Aggarwal, C.C. *Neural Networks and Deep Learning*; Springer: Cham, Switzerland, 2018; ISBN 978-3-319-94462-3.
23. Ding, H.; Wu, J.; Zhao, W.; Matinlinna, J.P.; Burrow, M.F.; Tsoi, J.K.-H. Artificial intelligence in dentistry—A review. *Front. Dent. Med.* **2023**, *4*, 1085251. [CrossRef]
24. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput.* **2019**, *31*, 1235–1270. [CrossRef]
25. Bharti, R.; Khamparia, A.; Shabaz, M.; Dhiman, G.; Pande, S.; Singh, P. Prediction of heart disease using a combination of machine learning and deep learning. *Comput. Intell. Neurosci.* **2021**, *2021*, 8387680. [CrossRef] [PubMed]
26. Aggarwal, A.; Mittal, M.; Battineni, G. Generative adversarial network: An overview of theory and applications. *Int. J. Inf. Manag. Data Insights* **2021**, *1*, 100004. [CrossRef]
27. Montesinos López, O.A.; Montesinos López, A.; Crossa, J. *Multivariate Statistical Machine Learning Methods for Genomic Prediction*; Springer Nature: Berlin/Heidelberg, Germany, 2022.
28. Bengio, Y.; Lecun, Y.; Hinton, G. Deep learning for AI. *Commun. ACM* **2021**, *64*, 58–65. [CrossRef]
29. Higham, C.F.; Higham, D.J. Deep learning: An introduction for applied mathematicians. *Siam Rev.* **2019**, *61*, 860–891. [CrossRef]
30. Molla, D.M.; Badis, H.; George, L.; Berbineau, M. Software defined radio platforms for wireless technologies. *IEEE Access* **2022**, *10*, 26203–26229. [CrossRef]
31. Ettus, C. Building_and_Installing_the_USRP_Open-Source_Toolchain_(UHD_and_GNU_Radio)_on_Linux. 2019. Available online: https://kb.ettus.com/Building_and_Installing_the_USRP_Open-Source_Toolchain_(UHD_and_GNU_Radio)_on_Linux (accessed on 20 May 2023).

32. Partiansyah, F.H.; Kusmaryanto, S.; Ambarwati, R.; Pramono, S.H. Experimental Study of USRP N210 as Simple GSM OpenBTS 5.0 for Remote Areas. In Proceedings of the 2022 11th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), Malang, Indonesia, 23–25 August 2022; pp. 185–190.
33. Esmaeily, A.; Kralevska, K. Small-scale 5g testbeds for network slicing deployment: A systematic review. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 6655216. [CrossRef]
34. Chica-Pedraza, G.; Mojica-Nava, E.; Cadena-Muñoz, E. Boltzmann distributed replicator dynamics: Population games in a microgrid context. *Games* **2021**, *12*, 8. [CrossRef]