applied sciences

MDPI

Article

Threat Detection Model for WLAN of Simulated Data Using Deep Convolutional Neural Network

Omar I. Dallal Bashi [1], Shymaa Mohammed Jameel [2], Yasir Mahmood Al Kubaisi [3], Husamuldeen K. Hameed [4] and Ahmad H. Sabry [5,*]

1 Medical Technical Institute, Northern Technical University, Mosul 41002, Iraq; omardallalbashi@ntu.edu.iq
2 Iraqi Commission for Computers and Informatics, Baghdad 10009, Iraq; shymaa792003@yahoo.com
3 Department of Sustainability Management, Dubai Academic Health Corporation, Oud Metha, Dubai 4545, United Arab Emirates; yaser.19711@gmail.com
4 Department of Telecommunications and Information, High Institute of Telecommunications and Post, Baghdad 10011, Iraq; husamuldeen72@gmail.com
5 Department of Computer Engineering, University of Al-Nahrain, Baghdad 64074, Iraq
* Correspondence: ahs4771384@gmail.com

Abstract: Security identification solutions against WLAN network attacks according to straightforward digital detectors, such as SSID, IP addresses, and MAC addresses, are not efficient in identifying such hacking or router impersonation. These detectors can be simply mocked. Therefore, a further protected key uses new information by combining these simple digital identifiers with an RF signature of the radio link. In this work, a design of a convolutional neural network (CNN) based on fingerprinting radio frequency (RF) is developed with computer-generated data. The developed CNN is trained with beacon frames of a wireless local area network (WLAN) that is simulated as a result of identified and unidentified router nodes of fingerprinting RF. The proposed CNN is able to detect router impersonators by comparing the data pair of the MAC address and RF signature of the received signal from the known and unknown routers. ADAM optimizer, which is the extended version of stochastic gradient descent, has been used with a developed deep learning convolutional neural network containing three fully connected and two convolutional layers. According to the training progress graphic, the network converges to around 100% accuracy within the first epoch, which indicates that the developed architecture was efficient in detecting router impersonations.

Keywords: WLAN; network security; network impersonations; MAC; IP address; ADAM optimizer

check for updates

Citation: Bashi, O.I.D.; Jameel, S.M.; Kubaisi, Y.M.A.; Hameed, H.K.; Sabry, A.H. Threat Detection Model for WLAN of Simulated Data Using Deep Convolutional Neural Network. *Appl. Sci.* **2023**, *13*, 11592. https://doi.org/10.3390/app132011592

Academic Editors: Lei Chen, Wenjia Li and Yun Lin

Received: 19 September 2023
Revised: 16 October 2023
Accepted: 20 October 2023
Published: 23 October 2023

1. Introduction

Wireless networks have surpassed wired networks in popularity. The demand for wireless networks stems from the fact that they provide enhanced accessibility and mobility, are adaptable and no extra infrastructure is required. The open nature of wireless networks opens the door to several security threats and breaches [1]. Analyzing wireless and mobile data is crucial for various purposes [2], with a strong emphasis on cybersecurity, risk management, attack detection, and crime analysis [3]. There are several cryptography-based technologies for wireless network authentication, data secrecy, and integrity. However, such tactics are rendered ineffective when faced with denial-of-service (DoS) assaults such as jamming. Furthermore, wireless security protocol implementations are known to be riddled with security flaws that may be readily exploited. For instance, passphrases in Wi-Fi Protected Access (WPA) and Wi-Fi Protected Access II (WPA2) [4,5], many more, man-in-the-middle attacks in cellular networks [6,7] and statistical analysis can jeopardize wired equivalent privacy (WEP). Finally, wireless sensor networks, many wireless networks, cognitive radio networks, including wireless mesh networks, and presuppose some level of user interaction, small cell networks. As a result, unique and low-complexity approaches for authenticating authorized users and detecting possible assaults from hostile adversaries are critical. Device silence technology has recently appeared, which is a technology that contributes to building device-specific signatures by collecting its information and using it in the process of identifying specific devices. This technology was used to reduce internal attacks or to reduce the wireless network's vulnerability to node falsification [6,8]. The fundamental idea is to actively or passively extract distinctive patterns (also known as features) expressed by target devices during wireless communication. An effective device fingerprint must satisfy two characteristics: (1) the features must be stable, especially in the presence of node movement and changes in the environment, and (2) They must be impossible or very difficult to counterfeit. The first requirement renders identifiers such as mobile identification number (MIN), international mobile station equipment identity (IMEI) number, electronic serial number (ESN), or IP address. Candidates are unsuitable because all of these IDs are easily changeable using software [9,10]. On the contrary, position-based features such as common radio signal strength (RSS) cannot be employed on their own as fingerprints because they are vulnerable to environmental and movement alters. Although most of the interests have focused on enhancing the wireless security and fingerprinting capabilities of wireless devices, it is surprising that there is no general insight in all of the current literature as the focus is sparse on the most recent fundamentals and key technologies involved.

Wireless network suffers from different conventional attacks like Denial of Service (DoS), DNS poisoning, Routing Information Protocol attack, Flooding, Address Resolution Protocol spoofing, IP spoofing, etc. A firewall and simple digital identifiers are a great defense against outside attacks, but it is ineffective against insider threats. Further, RF fingerprinting involves extracting unique features (or patterns) across the protocol stack that can be used as device signatures. Indeed, the upper layers, medium access control (MAC) layer and physical layer have been utilized for radio fingerprinting [11]. However, simple unique identifiers such as MAC addresses, IP addresses, and international mobile station equipment identity (IMEI) numbers can easily be spoofed. Location-based features such as channel state information (CSI), angle of arrival (AoA), and radio signal strength (RSS) are susceptible to environmental and mobility changes. The network architecture should have effective intrusion detection and prevention systems to lessen these threats. RF signature-based identification provides a more robust solution due to (1) low computational cost, (2) high detection precision for earlier known attacks, and (3) identifying intrusion by matching the preconfigured knowledge base with captured patterns [12].

Creating a Radio Frequency (RF) fingerprinting CNN for network security using simulated Wireless Local Area Network (WLAN) frames is a complex task that involves several steps [13]. This type of system can be used to detect unauthorized or rogue devices within a network, which might include the following points:

(1) Data Collection, Preparation, and Labeling: this is done by collecting a dataset of WLAN beacon frames and labeling the data, indicating whether each frame comes from an authorized or unauthorized device.
(2) Data Preprocessing: this step preprocesses the raw beacon frame data to extract relevant information, such as signal strength, MAC addresses, SSID (Service Set Identifier), and other features that can be used to identify devices. This stage involves converting and standardizing the data to make it suitable for input into a CNN.
(3) CNN Model Design: the design of a CNN architecture that is suitable for RF fingerprinting should take as input the preprocessed data and output a binary classification (authorized or unauthorized device). The network can have convolutional layers to extract relevant patterns from the RF data, followed by fully connected layers for classification.
(4) Model Training: This usually includes splitting the dataset into training and testing subsets, training the CNN on the training data, using binary classification (authorized or unauthorized) as the target, and using appropriate loss functions and optimization techniques for training.

(5) Model Evaluation: Evaluating the model's performance on the testing dataset is a common metric for classification problems including accuracy, precision, recall, and F1-score. In addition, this includes fine-tuning the model and hyperparameters to optimize performance.
(6) Real-Time Monitoring: this needs to capture and process live WLAN beacon frames in real-time, and feed these frames into the trained CNN for continuous monitoring.
(7) Alerting and Response: If the CNN detects a device as unauthorized, the model generates an alert or takes appropriate action. This could involve disconnecting the unauthorized device from the network or notifying network administrators.
(8) Regular Updates: this stage periodically updates the CNN model as the network evolves, and new devices are added. Re-train the model with fresh data to maintain its accuracy.
(9) Security Measures: In addition to using an RF fingerprinting CNN, employ other security measures like encryption, strong authentication, and intrusion detection systems to enhance network security.
(10) Compliance: this is to ensure that the network monitoring system complies with relevant regulations and privacy considerations, especially if we are handling sensitive information.

Huang et al. [14] addressed the constellation-error parameter using the SDA feature extraction classification method. The RF source was seven TDMA satellite terminals, and the study obtained 95% identification accuracy. Candore et al. [15] discussed the parameters of frequency offset, modulation phase offset, in-phase/quadrature-phase offset, and magnitude using the weighted voting-based classifier. The RF emitter was Six WARP radio cards. The results demonstrated acquired 88% identification accuracy and a 12.8% false alarm rate. The study [16] studied the IQ Imbalance parameter using SVM classification with MATLAB to simulate 5 analog modulators. The accuracy was $\geq$90% for SNR $\geq$ 15 dB. Brik et al. [7] discussed frequency error, SYNC correlation, IQ offset, magnitude error, and phase error as radiometric parameters, using k-NN & SVM classification techniques, with 138 802.11 NICs RF emitters, the accuracy was 99.9% for SVM and97% for k-NN.

The study [17] proposed a method for device-type and physical device categorization described by "GTID" depending on artificial neural networks to exploit differences in hardware compositions and clock skews of the network items. However, choosing the best set of features is difficult because so many different traits are used. When there are many devices, this also results in scalability issues, which raises the computational complexity of training. Deep neural networks offer prevailing frameworks for greatly increasing the number of layers and their neurons, leveraging large datasets, and learning complex functions. Authors of both [18,19] applied deep neural networks at the physical layer, particularly spotlighting Convolutional Neural Network (CNN) and IQ samples-based modulation recognition to categorize 11 dissimilar modulation formats. However, these studies only recognized the modulation type of transmitters and didn't recognize router impersonation.

Several related works have been done on deep learning-based device identification to identify a device in a WLAN based on its RF fingerprinting of transmitted signals. There are a number of different security aspects and applications. We can classify these studies into supervised and unsupervised deep neural networks. The unsupervised deep learning networks depend on a real-time grouping of samples as discussed in [20], where a non-parametric Bayesian method was used for device detection, while [21] depends on an infinite hidden Markov random field (iHMRF) model. In contrast, the supervised deep learning methods are based on a priori labeling of samples and can be further classified into similarity-based and classification-based techniques. The similarity-based method uses the matching concept with database entries, where the study [22] used the 802.11 wireless driver fingerprinting model, and [23] proposed data rate fingerprinting to attain inactive localization. The supervised classification-based models relied on unique class identification and can be sub-classified into conventional (handcraft feature extraction) [24,25] with frequency domain approach [7], with PARADIS fingerprinting, and [17] with GTID fingerprinting. Deep learning (multi-layer neural network) has been discussed by [26], which is based on modulation recognition CNN, and [27] with Deep learning of the physical layer. We are interested in researching hardware characteristics that are built into a device and that are also difficult and constant for malicious agents to duplicate. Deep CNN was applied at the physical layer in references [18,28], with a focus on modulation recognition. The studies categorized eleven various modulation systems. However, this method merely identifies the transmitter's modulation type, not a specific device as we do here.

This paper's major contribution is the development of a convolutional neural network (CNN) architecture [29,30] for radio frequency (RF) fingerprinting utilizing computer-produced data. We employed simulated WLAN beacon frames for training the created CNN from unknown and known RF fingerprint routers. Then, we compared the RF fingerprint indicated by the CNN with the MAC (Media Access Control) address of the received signals, and we combined an advanced deep-learning convolutional neural network with the ADAM optimizer to detect the WLAN router intrusions. Adam optimizer [31–33] is considered in this work as it merges the benefits of RMSProp [34] and AdaGrad [35] optimization techniques, which actively adapts the exponential decline rate for the first and second moment measures for updating parameters. In addition, Adam's algorithm is suitable for large amounts of data sparse gradients and non-stationary objective optimization with noise. Many parameters are frequently present in deep neural networks. Since most loss functions used in deep learning are convex functions [36], finding the global optimal solution is simple with Adam's optimization of the deep learning model and leads to the best convergence.

2. Materials and Methods

The hypothesis is that the network is able to detect an impersonating device via RF signals without using the layers of the protocol and stack for other more enhanced security mechanisms to do this at the physical layer itself. Therefore, it is possible that the indeed each device has a unique characteristic that imparts those characteristics onto the signal. If we learn the characteristics of those transmitted signals we will know who the device is. Hence, the method is broadly described as RF fingerprinting to detect these fingerprints or these characteristic signatures inside a radio's electromagnetic transmitted signals. Further, the main concept for identifying network devices underlying radio fingerprinting is to find distinctive features (patterns) and utilize them as device signatures. For radio fingerprinting, a number of features at the upper layers, medium access control (MAC), and physical (PHY) layer have been used [3]. International mobile station equipment identity (IMEI) numbers, MAC addresses, and IP addresses are examples of straightforward unique identifiers that are simple to spoof. Environmental and mobility changes can affect location-based aspects including channel state information (CSI) and radio signal strength (RSS).

The representativeness and balance of the dataset are critical considerations when working with deep-learning neural networks. A dataset that is both representative and balanced is essential for training a model that generalizes well to real-world data. Further, a representative and balanced dataset is fundamental to the success of deep learning models. It ensures that the model can generalize well to unseen data, make accurate predictions, and avoid biases or skewed performance. Balancing classes, data augmentation, and thoughtful dataset curation are essential steps to achieve this. Here are some key aspects to consider:

- Dataset Size: The dataset should be large enough to cover the diversity of real-world scenarios. A small dataset may result in overfitting, where the model learns to perform well on the training data but fails to generalize to unseen data. In this work, we generated a 5000 Non-HT dataset of WLAN beacon frames for each router.
- Class distribution: Simulating a static Rayleigh fading channel with a specific delay profile and average path gains effectively applies a known transformation to the data. This introduces variations in the received signal characteristics while keeping the channel static. The fading channel effectively acts as a data augmentation technique to

create diversity within the dataset. By introducing variations, we generate additional samples for the minority class and improve the performance of the deep-learning model when dealing with class imbalance.

2.1. RF Signature-Based Router Impersonation Detection

In order to mislead network users into connecting to it, a malicious agent will try to impersonate a genuine router in a technique known as router impersonation. Simple digital identifiers used in security identification systems, including MAC addresses, IP addresses, and SSIDs, are ineffective at identifying such attacks. These detectors can be easily tricked. Consequently, additional secured solutions use further communication data, like the radio RF link fingerprinting combined with these straightforward digital detectors.

A receiver-to-transmitter wireless pair produces a single RF signature from the receiver for combining the RF impairments and channel. The process of identifying transmitted RF signals in a shared spectrum throughout these signatures is called Fingerprinting RF. The study [37] presented a design of a deep learning architecture to consume raw baseband in quadrature/phase (IQ) to identify and sample the signal-transmitting radios. The architecture can recognize an RF transmitting radio if the channel profiles stay steady through the processing time or the radio impairments are dominant. The majority of WLAN architectures have permanent routers for creating a stationary channel signature if the receiver position is also constant. For this scenario, the deep neural networks are able to define router impersonators by evaluating the data pair of the MAC address and the received signal's RF fingerprint to that of the identified routers.

These works create several fixed routers for the WLAN system with a permanent observer utilizing the WLAN Toolbox and perform neural network training on the computer-generated data by Deep Neural Learning. This approach assumes the following environment:

1. The network includes several trusted known routers (of identified MAC addresses) operating in an indoor area.
2. Some unidentified routers may include router intrusions within the inspection network coverage.
3. The "Unknown" category stands for each transmitting apparatus that is not included within the known group of routers.

Figure 1 demonstrates the three known router scenarios.

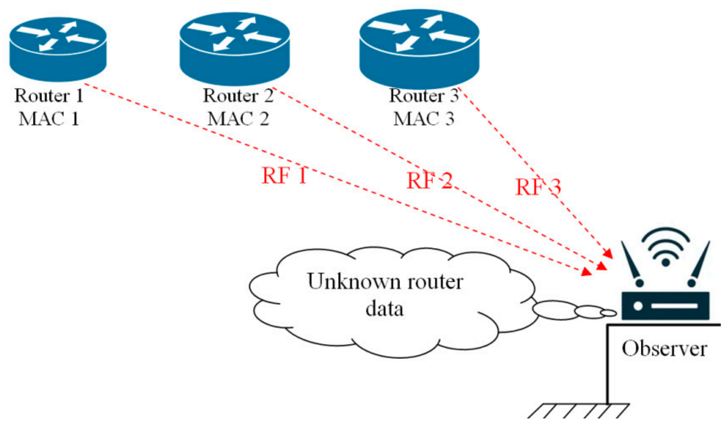


Figure 1. The diagram of the first scenario of three known routers.

The observer node gathers lighthouse signals of the routers with non-HT (non-high throughput) and identifies the RF fingerprint using the long (legacy) training field (L-LTF). To avoid any data dependency, L-LTF Transmitted signals are configured as the same for every router to allow algorithm application. Since the observer and the routers are constant, the RF signatures (combination of RF impairments and multi-path channel pattern) RF3, RF2, and RF1 perform without variation with time. Unidentified router information is a set of random RF signatures that are unlike the identified routers.

In another scenario in which a user is linked to a mobile hot spot and a router, the observer decodes the MAC address and collects beacon frames after training. Then, the observer takes out the L-LTF signals and employs these signals to categorize the RF signature for the beacon frame source. The observer assigns that the source is a "known router" if the RF fingerprint and the MAC address match, as shown for Router 3, Router 2, and Router 1. In the same context, the observer defines the source as an "unknown router" if the RF signature does not agree with every one of the identified routers and the beacon MAC address is not included in the database, as is demonstrated for a mobile hot spot in Figure 2.

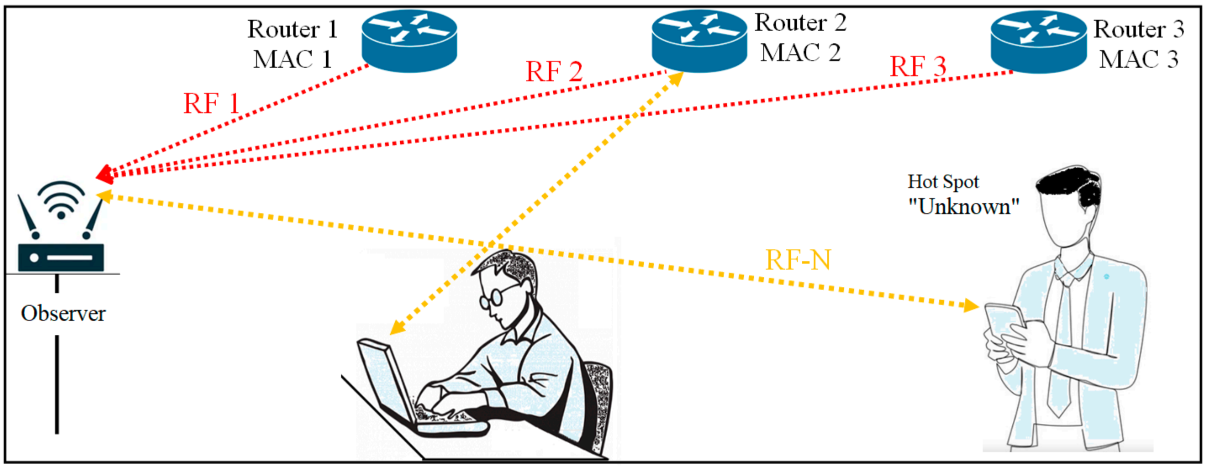


Figure 2. A user is linked to a mobile hot spot and a router, where the observer decodes the MAC address and collects beacon frames after training.

The third scenario demonstrates the effect of a router intrusion; where an impersonator router (evil twin) tries to transmit beacon frames by replicating the MAC address of an identified router, see Figure 3. The original router can be jammed by the hacker, forcing users to connect to its evil duplicate. The observer decodes the MAC address when receiving the beacon frames from the evil duplicate. The MAC address of an identified router matches the decoded MAC address but the RF signature does not counterpart. Then, the data source is identified by the observer as a router impostor.

2.2. System Parameters Setting

1. For each router, we generated a 5000 Non-HT dataset for WLAN beacon frames.
2. For the known routers, we used the router MAC addresses as known labels, while the remaining are "Unknown" labels.
3. The developed CNN is trained to detect any unknown routers and to categorize the known routers.
4. We divide the data into a test of 10%, validation also of 10%, and training of the remaining 80%.
5. The signal-to-noise ratio (SNR) is considered to work on a 5 GHz band and 20 dB.
6. The numbers of known and unknown devices are flexible. In this paper, the simulation applies two scenarios: 4 known against 10 unknown devices first and 7 known against 3 unknown devices the second time, with the ability to increase or decrease these numbers.
7. The number of routers that are marked as "unknown" is assumed to be greater and less than that for the known devices to model the variability in the dataset for the RF signatures of the unknown router. Table 1 shows the set values of the model parameters.

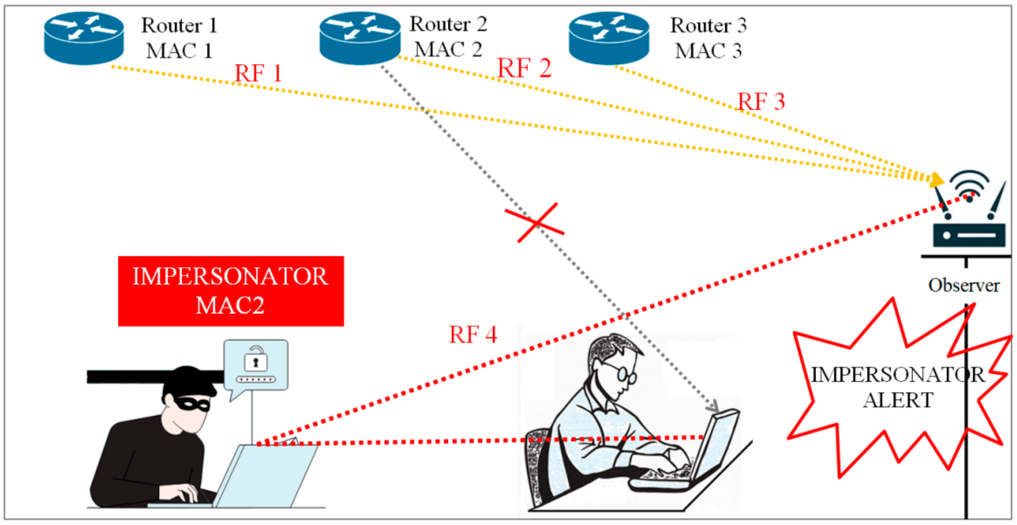


Figure 3. The effect of a router intrusion is that an impersonator router (evil twin) tries to transmit beacon frames by replicating the MAC address of a known router.

Table 1. The setting values of the model parameters.

Parameter	**Value**	**Parameter**	**Value**
No. of Known Routers	4, and 7	Channel Number	153 (WLAN channel number)
No. of Unknown Routers	10, and 3	Channel Band	5 (GHz)
No. of Total Routers	No. Known Routers + No. of Unknown Routers	Frame Length	160 (L-LTF sequence length in samples)
SNR	20 dB	No. of total Frames Per Route	5000
No. of training Frames Per Router	No. of Total Frames Per Router × 0.8	No. of test Frames Per Router	No. of Total Frames Per Router × 0.1;
No. of Validation Frames Per Router	No. of Total Frames Per Router × 0.1		

For the considered dataset, the network training taking longer than 3 min with an Intel® Xeon® @ 3.6 GHz W-2132CPU and about 34 s NVIDIA® 3080 GPU GeForce RTX. The production of 5000 frames consumes about 153 min on a computer of W-2133CPU @ 3.66 GHz Intel® Xeon® with 32 GB memory.

2.3. WLAN Waveforms Generation

Routers implementing 802.11a/g/ac Wi-Fi protocols send beacon brackets within 5 GHz bands to disclose their existence and abilities to use the non-HT OFDM layout. The frame of the beacon includes two common elements: payload (DATA) and preamble (SYNC), which in turn has two elements: long and short training.

This work includes a payload with the same bits excluding every router's MAC address. The CNN employs training units for the L-LTF element of the preamble. We perform reprocessing the RF signature for the L-LTF signal to provide an overhead-free signature explanation. To produce beacon frames for the WLAN the steps are implemented according to the flowchart in Figure 4.

2.4. RF Impairments and Channel Configuration

This stage includes passing every frame throughout two channels: (1) AWGN and Rayleigh multipath fading, and (2) Radio impairments, which may contain DC offset; frequency offset, and phase noise.

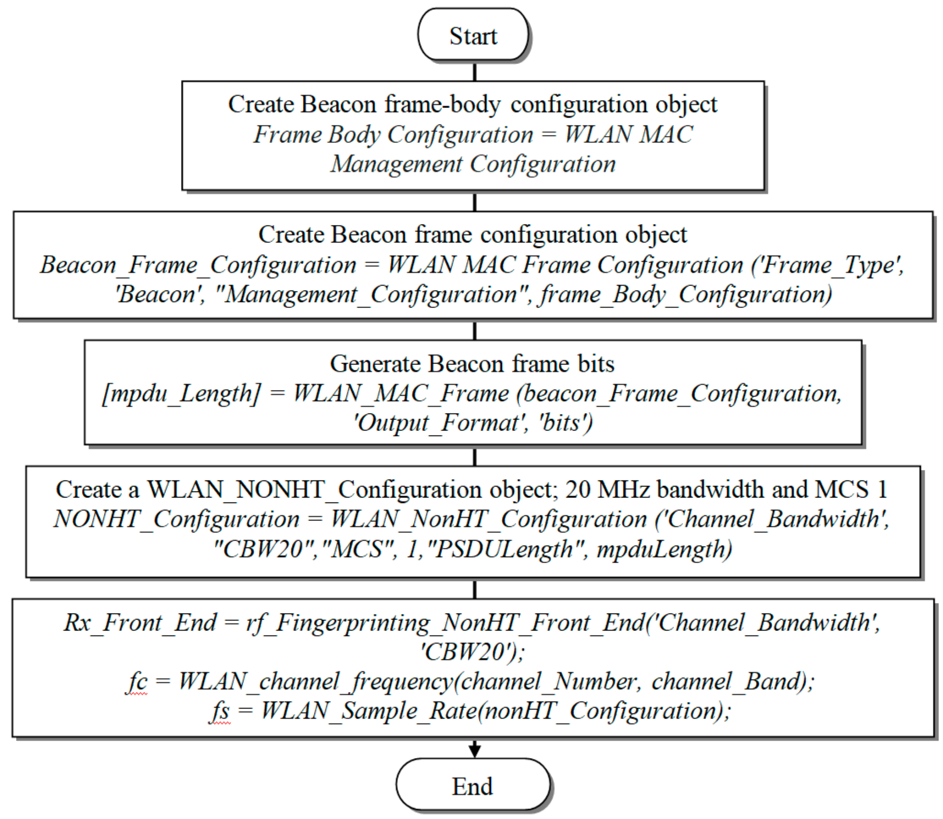

Start
Create Beacon frame-body configuration object
Frame Body Configuration = WLAN MAC Management Configuration
Create Beacon frame configuration object
Beacon_Frame_Configuration = WLAN MAC Frame Configuration ('Frame_Type', 'Beacon', "Management_Configuration", frame_Body_Configuration)
Generate Beacon frame bits
[mpdu_Length] = WLAN_MAC_Frame (beacon_Frame_Configuration, 'Output_Format', 'bits')
Create a WLAN_NONHT_Configuration object; 20 MHz bandwidth and MCS 1
NONHT_Configuration = WLAN_NonHT_Configuration ('Channel_Bandwidth', "CBW20","MCS", 1,"PSDULength", mpduLength)
Rx_Front_End = rf_Fingerprinting_NonHT_Front_End('Channel_Bandwidth', 'CBW20');
fc = WLAN_channel_frequency(channel_Number, channel_Band);
fs = WLAN_Sample_Rate(nonHT_Configuration);
End

Figure 4. Beacon frames production for the WLAN.

2.4.1. Rayleigh Multipath and AWGN

We assume an average pathway gain of [0, −2, −10] dB with corresponding delay pattern samples of [0 1.81 3.41]. The signals are sent through a Rayleigh multipath fading and static channel. Therefore, to ensure that there is no change in channel for the same RF, we set the upper limit of Doppler shift to zero. Then we apply these settings with the multipath channel and add *awgn* noise such that: Path Delays = [0 1.81 3.41]/fs, Average Path Gains = [0, −2, −10], and the Maximum Doppler Shift = 0.

Simulating a static Rayleigh fading channel with a specific delay profile and average path gains effectively applied a known transformation to the data. This introduced variations in the received signal characteristics while keeping the channel static. The fading channel effectively acted as a data augmentation technique to create diversity within the dataset. By introducing variations, we've generated additional samples for the minority class and improved the performance of the deep-learning model when dealing with class imbalance.

2.4.2. RF Impairments

The range of values and their corresponding radio impairments are DC offset = [−50, −32] dBc, Frequency offset = [−4, 4] ppm, and Phase noise = [0.01, 0.3] degrees (RMS). Then, we allocate arbitrary impairments to every simulated RF contained by the earlier known ranges as in Algorithm 1.

Algorithm 1. RF impairments

```
% each simulated radio is assigned with random impairments within the earlier defined ranges
Radio_Impairments = repmat (struct ('Phase_Noise', 0, 'DC_Offset', 0, 'Frequency_Offset', 0), ...
  Num_Total_Routers, 1);
for router_Idx = 1:num_Total_Routers
    radio_Impairments (router_Idx).Phase_Noise = rand*(phase_Noise_Range (2)-phase_Noise_Range (1)) +
  phase_Noise_Range (1);
    radio_Impairments (router_Idx).DC_Offset = rand*(dc_Offset_Range (2)-dc_Offset_Range (1)) + dc_Offset_Range (1);
    radio_Impairments (router_Idx).Frequency_Offset = fc/1e6*(rand*(freq_Offset_Range (2)-freq_Offset_Range (1)) +
  freq_Offset_Range (1));
end
```

2.5. Generating Data Frames for Training and Applying Channel Impairments

The generated data includes the generation of packets holding MAC beacon frames, which is a kind of managing frames to identify a basic service set (BSS) produced by several 802.11 devices appropriate for baseband modeling. A platform called software-defined radio (SDR) was used according to Section 9 IEEE standard [38]. Beacon frames consist of; (1) a valid frame check sequence (FCS), (2) a beacon frame body, which carries data that requests to transmit, and (3) a MAC header, which contains frame information. The transmitter computes the FCS over the frame body and header. The receiver employs the FCS to ensure that the frame body and header are properly created.

An experimental setup using USRP SDRs is conducted to acquire I/Q samples as shown in Figure 5. We use several different devices with USRP B210 family as transmitter devices, while a fixed USRP B210 is used for data collection at the receiver end.

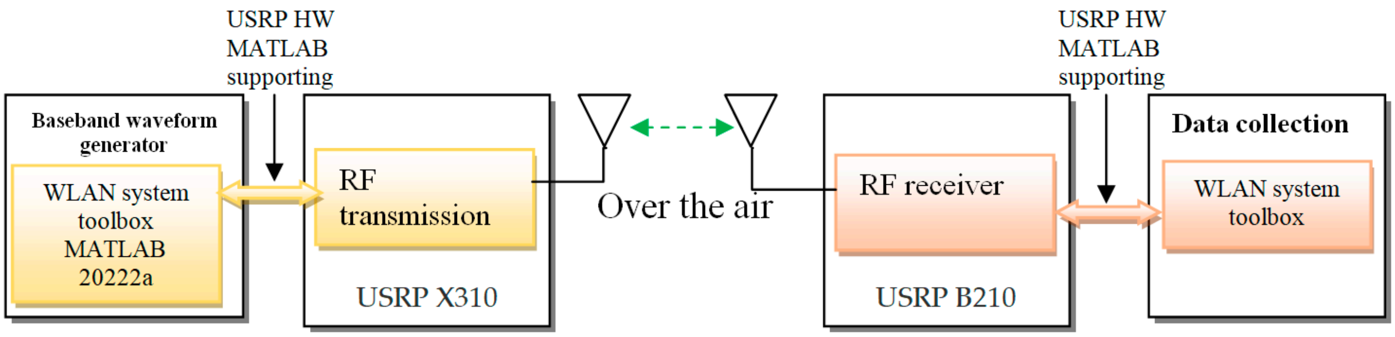


Figure 5. SDR-based Data collection.

On each transmitter SDR, we send several physical layer frames according to IEEE 802.11ac standards. These frames are compliant with standards and produced using the MATLAB WLAN Systems toolbox.

Since we want to communicate any data streams, the data frames that are generated are random. The chosen SDR receives these protocol frames and streams them for over-the-air wireless transmission. For WiFi, the receiving SDR samples the inbound signals at a central frequency of 2.45 GHz and at a rate of 1.92 MS/s. Complex I/Q samples are gathered and divided into subsequences. We used a fixed subsequence length of 128 for our experimental study; further information is provided below. For each of the five SDRs, we gathered roughly 20 million samples in total. A general MAC frame diagram is shown in Figure 6 [39].

Generating MAC beacon frames includes three main stages; the control frames (Block Ack, Ack, CTS, and RTS), data frames (QoS Null, QoS data, Null, Data), and management frames (beacon). The flow work stages of generating MAC beacon frames are demonstrated in Figure 7.

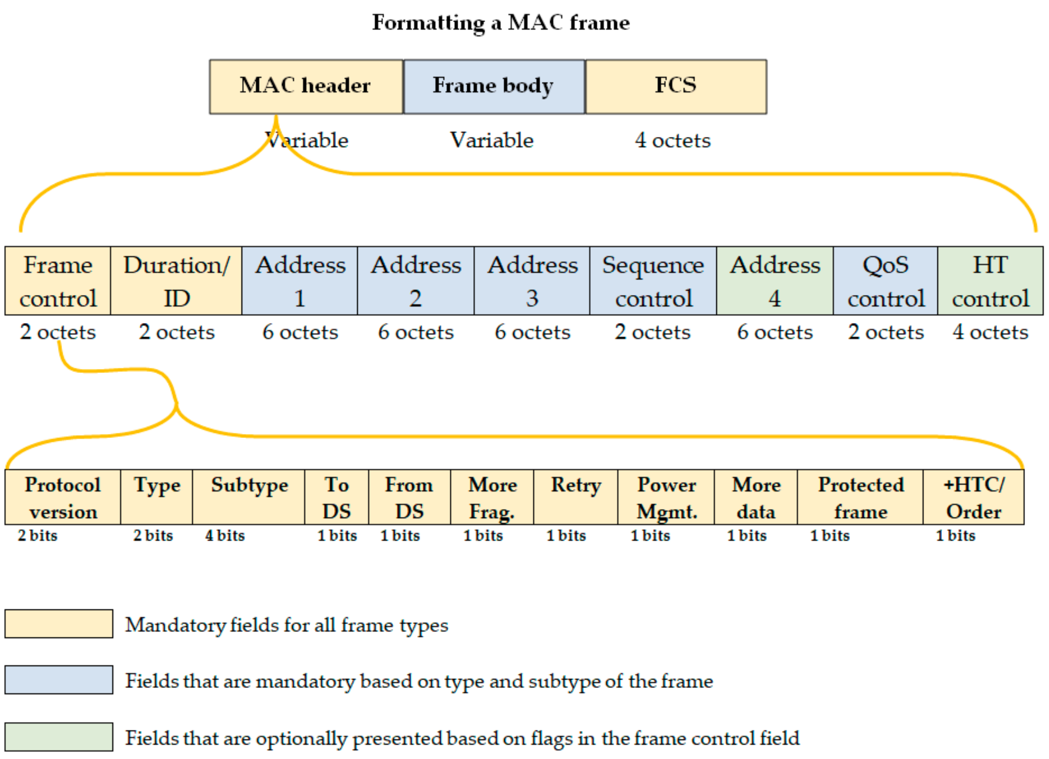
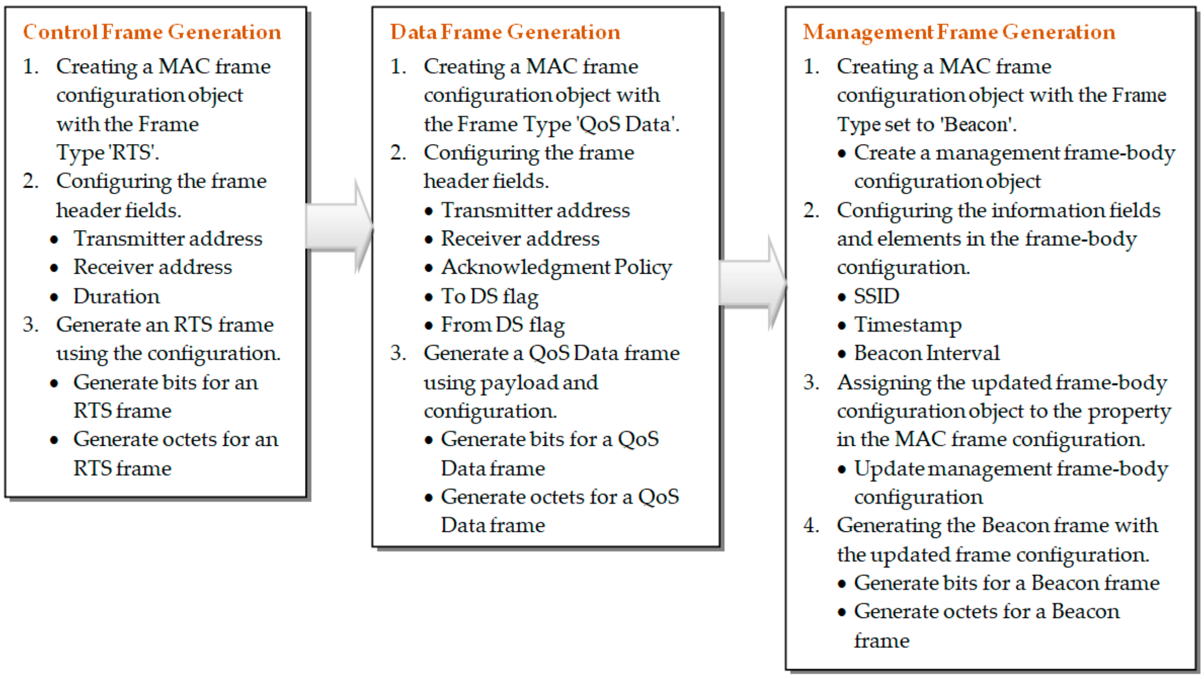


Figure 6. A general MAC frame diagram.



Control Frame Generation

1. Creating a MAC frame configuration object with the Frame Type 'RTS'.
2. Configuring the frame header fields.
   - Transmitter address
   - Receiver address
   - Duration
3. Generate an RTS frame using the configuration.
   - Generate bits for an RTS frame
   - Generate octets for an RTS frame

Data Frame Generation

1. Creating a MAC frame configuration object with the Frame Type 'QoS Data'.
2. Configuring the frame header fields.
   - Transmitter address
   - Receiver address
   - Acknowledgment Policy
   - To DS flag
   - From DS flag
3. Generate a QoS Data frame using payload and configuration.
   - Generate bits for a QoS Data frame
   - Generate octets for a QoS Data frame

Management Frame Generation

1. Creating a MAC frame configuration object with the Frame Type set to 'Beacon'.
   - Create a management frame-body configuration object
2. Configuring the information fields and elements in the frame-body configuration.
   - SSID
   - Timestamp
   - Beacon Interval
3. Assigning the updated frame-body configuration object to the property in the MAC frame configuration.
   - Update management frame-body configuration
4. Generating the Beacon frame with the updated frame configuration.
   - Generate bits for a Beacon frame
   - Generate octets for a Beacon frame

Figure 7. Flow work stages of generating MAC beacon frames.

This stage consists of applying the channel impairments and RF defined before, generating an independent channel by resetting the channel object for each radio, processing the received frames using *RF_FingerprintingNonHT_Front_End* function, performing L-LTF extraction from each WLAN transmitted frame, and splitting the L-LTF received signal data into test, validation and training parts. This process can be demonstrated in a flowchart shown in Figure 8.

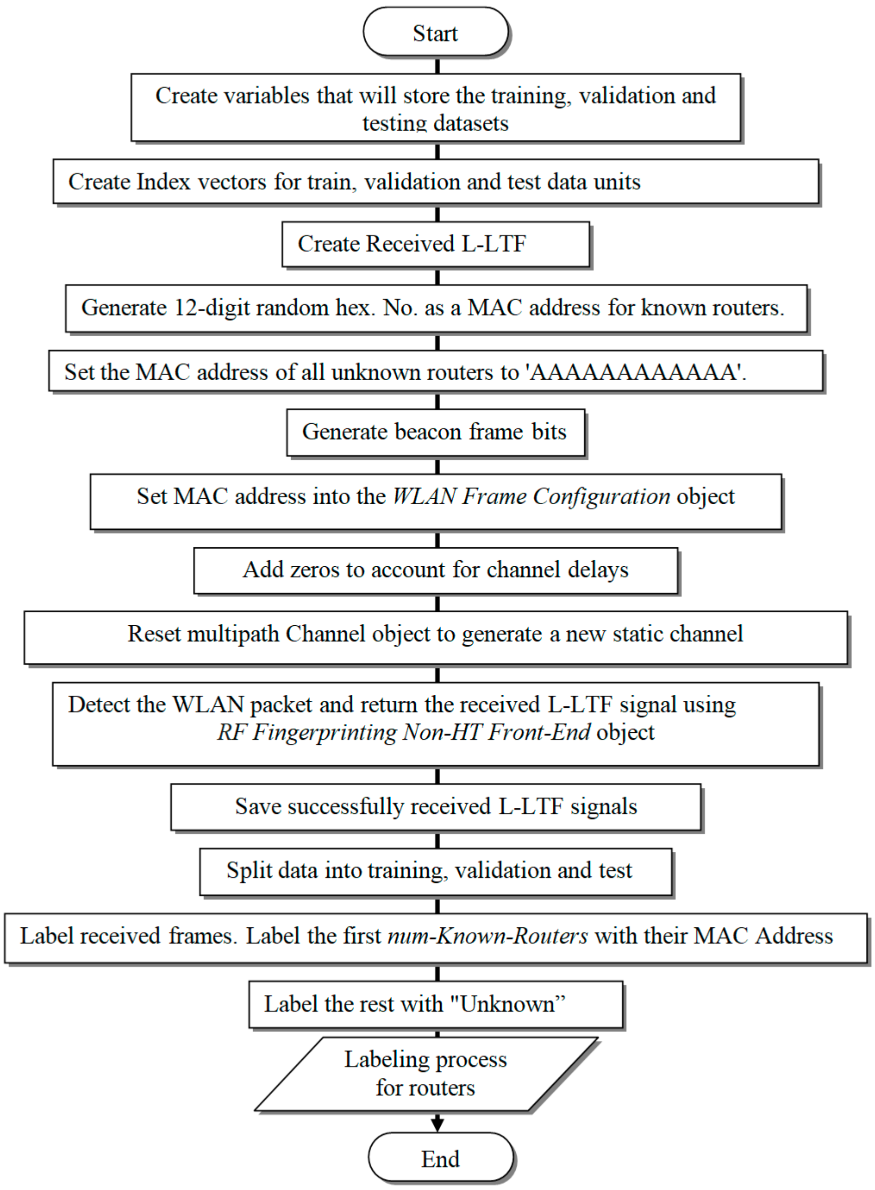

Start
Create variables that will store the training, validation and testing datasets
Create Index vectors for train, validation and test data units
Create Received L-LTF
Generate 12-digit random hex. No. as a MAC address for known routers.
Set the MAC address of all unknown routers to 'AAAAAAAAAAAA'.
Generate beacon frame bits
Set MAC address into the *WLAN Frame Configuration* object
Add zeros to account for channel delays
Reset multipath Channel object to generate a new static channel
Detect the WLAN packet and return the received L-LTF signal using *RF Fingerprinting Non-HT Front-End* object
Save successfully received L-LTF signals
Split data into training, validation and test
Label received frames. Label the first *num-Known-Routers* with their MAC Address
Label the rest with "Unknown"
Labeling process for routers
End

Figure 8. Generating Data Frames for Training and Applying Channel Impairments.

2.6. Creating Input Matrices with Real-Values

Real numbers are necessary for performing the developed deep learning network. Therefore, I/Q data is used as a representation of how an RF carrier is modulated in frequency, amplitude, and phase. The terms “in-phase (I)” and “quadrature (Q)” refer to the two amplitude-modulated sinusoids’ interactions with the carrier’s amplitude- and phase-modulated signal, respectively. Therefore, Q and I are separated into two individual vectors so that the deep learning network can deal with them, which is necessary to create real-valued input matrices. In addition, we apply a mixing between the training dataset and saving the variable of classes as conclusive data. This process can be demonstrated in a flowchart shown in Figure 9.

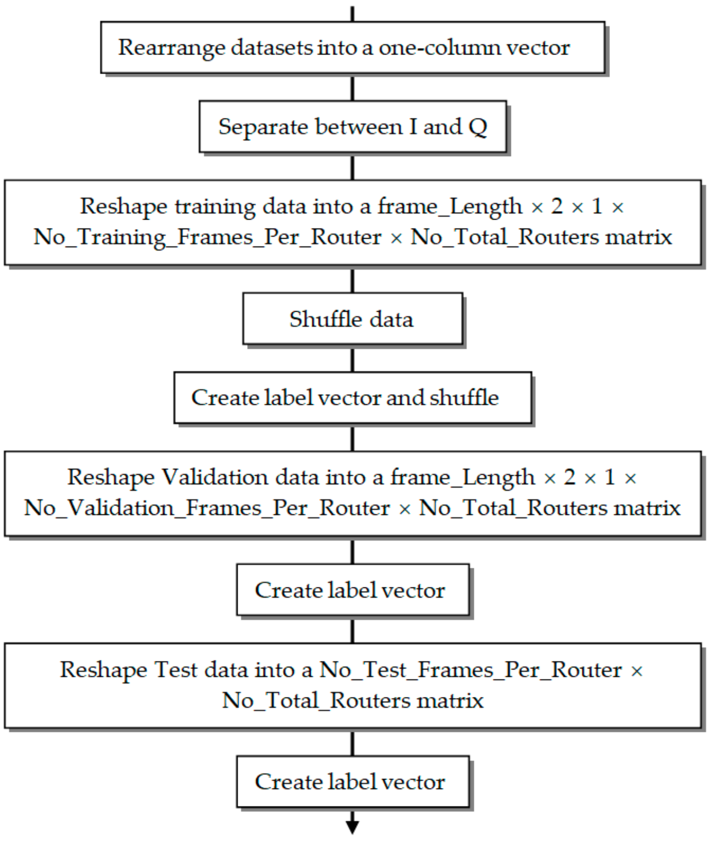

Rearrange datasets into a one-column vector
Separate between I and Q
Reshape training data into a frame_Length × 2 × 1 × No_Training_Frames_Per_Router × No_Total_Routers matrix
Shuffle data
Create label vector and shuffle
Reshape Validation data into a frame_Length × 2 × 1 × No_Validation_Frames_Per_Router × No_Total_Routers matrix
Create label vector
Reshape Test data into a No_Test_Frames_Per_Router × No_Total_Routers matrix
Create label vector

Figure 9. Creating Input Matrices with Real-Values.

2.7. The Developed Neural Network Architecture

IQ imbalance has been considered in this work due to that the quadrature mixers are usually degraded by the mismatch in phase and gain between the parallel RF chain branches handling the Q and I signal tracks. Phase imbalance comes from the quadrature signal's phase divergence from 90 degrees, whereas amplitude imbalance is caused by the mismatch in their gains. Due to frequency-dependent low-pass filters, IQ imbalance only fluctuates with frequency, and as a result, it bears a particular transmitter signature for that frequency.

This work develops a deep learning convolutional neural network (CNN) that contains three fully connected and two convolutional layers. The training options have been configured to use a mini-batch size of 512 with the ADAM optimizer, which is the extended version of stochastic gradient descent and can be implemented in natural language and computer vision processing. The perception behind this model is to learn features independently by the first layer in Q and I. The filter size is 1×7, while the size is 2×7 at the next layer for extracting features that combine both Q and I. Lastly; the extracted features in the previous layers are classified by a final three fully connected layers [37]. To make sure that the applied CNN model generalizes successfully, the selection of network hyperparameters, such as the number of filters and their sizes in the convolution layers, and CNN depth, is crucial. These are carefully selected via cross-validation. We fixed the dropout rate for the dense layers to 50% in order to avoid overfitting. Additionally, we choose a regularization factor of = 0.0001. Adam optimizer is used to train the network's weights, and its learning rate is set to lr = 0.0001. Back-propagation is used to minimize error prediction, with categorical cross-entropy computed on the classifier output serving as the loss function. The developed architecture is detailed in Table 2.

Table 2. The layers of the developed architecture.

Name/Size	Description	Layer Label	No
160 × 2 × 1 images	Image Input	'Input Layer'	1
Stride padding [1 1 0 0] and [1 1] with 50 7 × 1 convolutions	2-D Convolution	'CNN1'	2
Batch normalization	Batch Normalization	'BN1'	3
Leaky ReLU with a scale of 0.01	Leaky ReLU	'LeakyReLu1'	4
2 × 1 max pooling with stride [2 1] and padding [0 0 0 0]	2-D Max Pooling	'MaxPool1'	5
50 7 × 2 convolutions with stride [1 1] and padding [1 1 0 0]	2-D Convolution	'CNN2'	6
Batch normalization	Batch Normalization	'BN2'	7
Leaky ReLU with a scale of 0.01	Leaky ReLU	'LeakyReLu2'	8
2 × 1 max pooling with stride [2 1] and padding [0 0 0 0]	2-D Max Pooling	'MaxPool2'	9
256 fully connected layer	Fully Connected	'FC1'	10
Leaky ReLU with a scale of 0.01	Leaky ReLU	'LeakyReLu3'	11
50% dropout	Dropout	'DropOut1'	12
80 fully connected layer	Fully Connected	'FC2'	13
Leaky ReLU with a scale of 0.01	Leaky ReLU	'LeakyReLu4'	14
50% dropout	Dropout	'DropOut2'	15
5 fully connected layer	Fully Connected	'FC3'	16
Softmax	Softmax	'SoftMax'	17
Crossentropyex	Classification Output	'Output'	18

After setting the training options, network training is performed. Then, we loaded the trained network, testing dataset, and the user-generated MAC Addresses. The accuracy of the network architecture is computed, and the classification for the testing frames is performed to detect router imitators.

For each unknown MAC address and all the known MAC addresses in iteration, a generation of beacon frames is performed. Then, generating a new pack of multipath channels and RF impairments is performed. The RF profile for these frames should be categorized as "Unknown" because all of the impairments are unexplored. Beacon frames of defined MAC addresses represent router imitators whereas those of undefined MAC addresses are unknown devices. The impersonator detection algorithm can be represented by Algorithm 2.

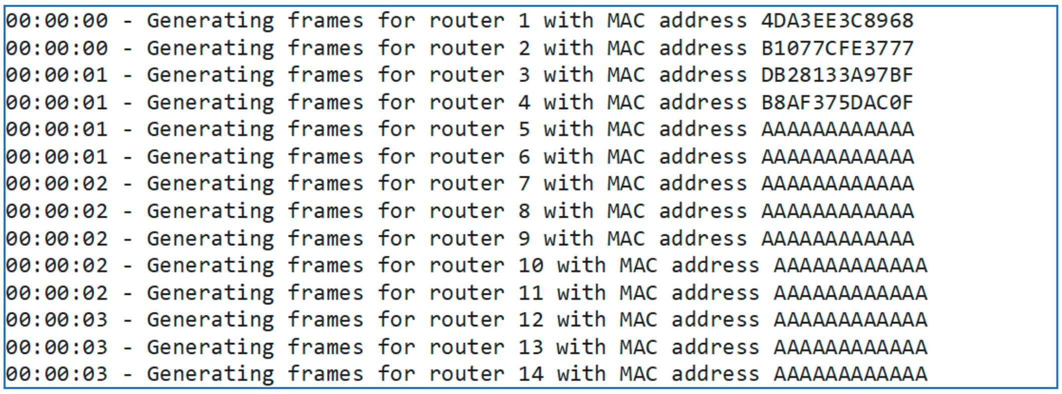
Algorithm 2. Impersonator detection algorithm

```
Frames_Per_Router = 4;
Known_MAC_Addresses = generated_MAC_Addresses (1: No_Known_Routers);
% each simulated radio is assigned as random impairments within the earlier defined ranges
For router_Idx = 1: No_Total_Routers
        radio_Impairments (router_Idx).Phase_Noise = rand*( phase_Noise_Range (2)-phase_Noise_Range (1) ) +
        phase_Noise_Range (1);
        radio_Impairments (router_Idx).DC_Offset = rand*( DC_Offset_Range (2)-DC_Offset_Range (1) ) + DC_Offset_Range (1);
        radio_Impairments (router_Idx).Frequency_Offset= fc/1e6*(rand*( freq_Offset_Range(2)-freq_Offset_Range(1) ) +
        freq_Offset_Range(1));
end
% To generate a new static channel, Reset multipath_Channel object
Reset (multipath_Channel)
% Do for one unknown routers and all known
For mac_Index = 1:(No_Known_Routers + 1)
   beacon_Frame_Config.Address2 = generated_MAC_Addresses(macIndex);
   % produce Beacon frame bits
   beacon = wlan_MAC_Frame(beacon_Frame_Config, 'Output_Format', 'bits');
   tx_Waveform = wlan_Waveform_Generator(beacon, nonHT_Config);
   tx_Waveform = helper_Normalize_Frame_Power (tx_Waveform);
   % To account, add zeros for channel delays
   tx_Waveform = [tx_Waveform; zeros (160,1) ];
   % generate unseen RF fingerprint or an undetected multipath channel.
   Reset (multipath_Channel)
   Frame_Count= 0;
   while frame_Count < frames_Per_Router
      rxMultipath = multipath_Channel(tx_Waveform);
      rxImpairment = helperRFImpairments(rxMultipath, radio_Impairments(router_Idx), fs);
      rxSig = awgn(rxImpairment,SNR,0);
      % Detect the WLAN packet and return the received L-LTF signal using rf_Fingerprinting_NonHT_Front_End object
    [payload_Full, cfgNonHT, rx_NonHT_Data, chanEst, noiseVar, LLTF] = rx_Front_End (rxSig);
    If payload_Full
    frame_Count = frameCount+1;
    rec_Bits = wlan_NonHT_Data_Recover (rx_NonHT_Data, chanEst, noise_Var, cfg_NonHT, 'Equalization_Method', 'ZF');
    % Evaluate and decode recovered bits
    mpduCfg = wlan_MPDU_Decode (recBits, cfgNonHT);
    % Reshape and Separate Q and I of neural network
    LLTF= [ real (LLTF), imag (LLTF) ];
    LLTF = permute (reshape (LLTF, frame_Length, [] ,   2,   1), [1 3 4 2]);
    ypred = classify(sim_Net, LLTF);
     if sum (contains (known_MAC_Addresses, mpduCfg.Address2 )) ~= 0
      if categorical (convert_Chars_To_Strings (mpduCfg.Address2))~=ypred
       disp (strcat ( " MAC Address ", mpduCfg.Address2," is known, fingerprint mismatch, ROUTER IMPERSONATOR
       DETECTED" ))
      else
       disp(strcat("MAC Address ", mpduCfg.Address2," is known, fingerprint match"))
      end
     else
        disp(strcat("MAC_Address ", mpduCfg.Address2," is not recognized, unknown device"))
     end
    end
    % To generate a new static channel, reset multipath_Channel object
    Reset (multipath_Channel)
   end
 end
```

3. Results

3.1. Scenario 1, Generating AA. . .A MAC Address for the Unknown Devices

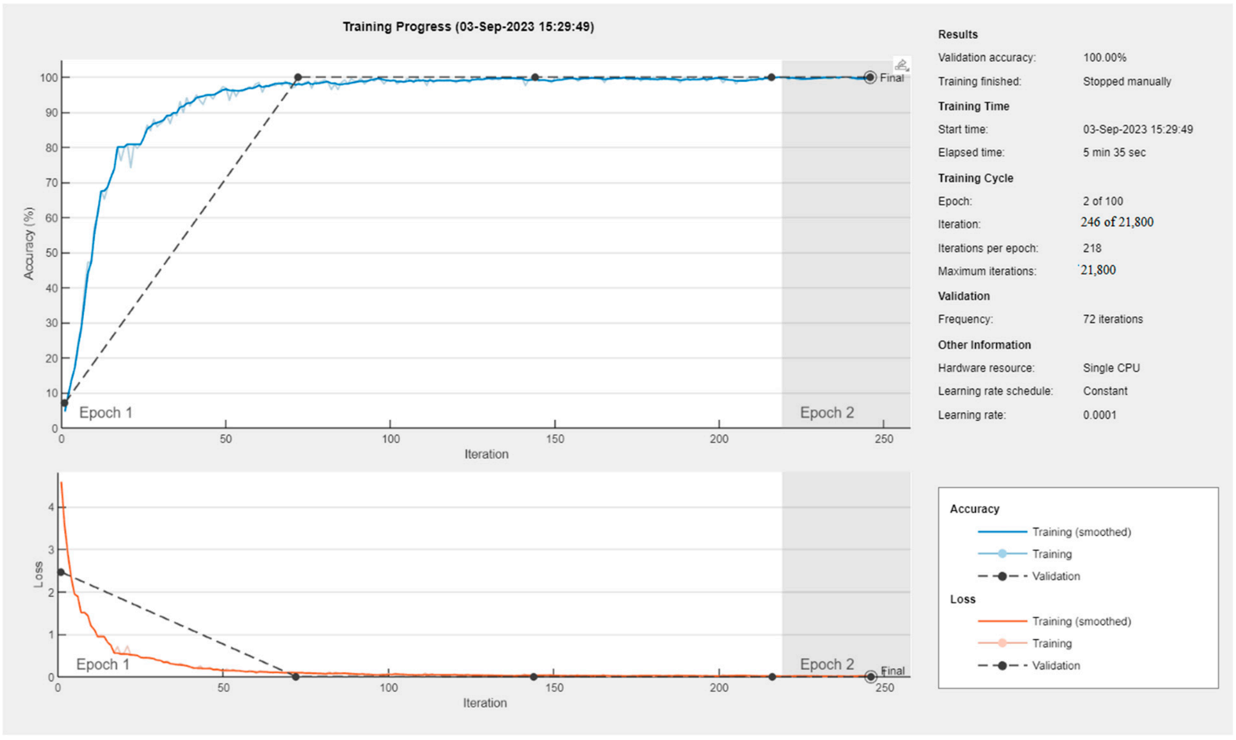
The operation of MAC address generation after applying channel impairments results in data frames for the router of the WLAN, a photo of the generated data is shown in Figure 10.



```
00:00:00 - Generating frames for router 1 with MAC address 4DA3EE3C8968
00:00:00 - Generating frames for router 2 with MAC address B1077CFE3777
00:00:01 - Generating frames for router 3 with MAC address DB28133A97BF
00:00:01 - Generating frames for router 4 with MAC address B8AF375DAC0F
00:00:01 - Generating frames for router 5 with MAC address AAAAAAAAAAAA
00:00:01 - Generating frames for router 6 with MAC address AAAAAAAAAAAA
00:00:02 - Generating frames for router 7 with MAC address AAAAAAAAAAAA
00:00:02 - Generating frames for router 8 with MAC address AAAAAAAAAAAA
00:00:02 - Generating frames for router 9 with MAC address AAAAAAAAAAAA
00:00:02 - Generating frames for router 10 with MAC address AAAAAAAAAAAA
00:00:02 - Generating frames for router 11 with MAC address AAAAAAAAAAAA
00:00:03 - Generating frames for router 12 with MAC address AAAAAAAAAAAA
00:00:03 - Generating frames for router 13 with MAC address AAAAAAAAAAAA
00:00:03 - Generating frames for router 14 with MAC address AAAAAAAAAAAA
```

Figure 10. Generating the data frames for routers.

There are four normally generated MAC addresses of the known routers, while A. . .A MAC address for the remaining 10 unknown routers. The training options have been configured to use a mini-batch size of 512 with the ADAM optimizer, which is the extended version of stochastic gradient descent and can be implemented in natural language and computer vision processing. According to the network specifications in Table 2, the training progress showing the accuracy and loss along with the iterations is shown in Figure 11.



Results

Validation accuracy:	100.00%
Training finished:	Stopped manually

Training Time

Start time:	03-Sep-2023 15:29:49
Elapsed time:	5 min 35 sec

Training Cycle

Epoch:	2 of 100
Iteration:	246 of 21,800
Iterations per epoch:	218
Maximum iterations:	21,800

Validation

Frequency:	72 iterations

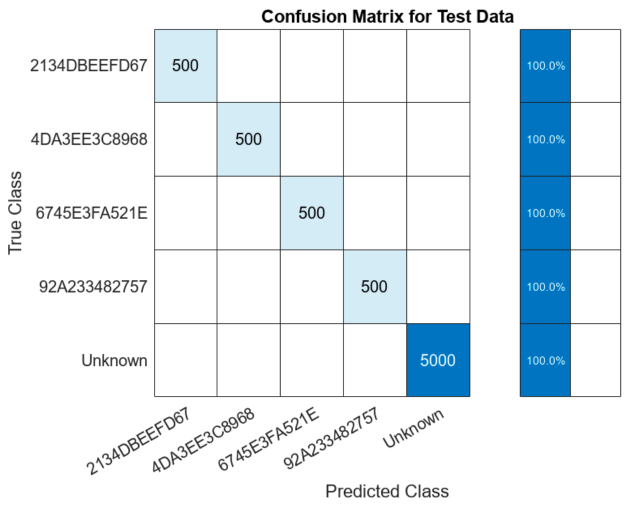
Other Information

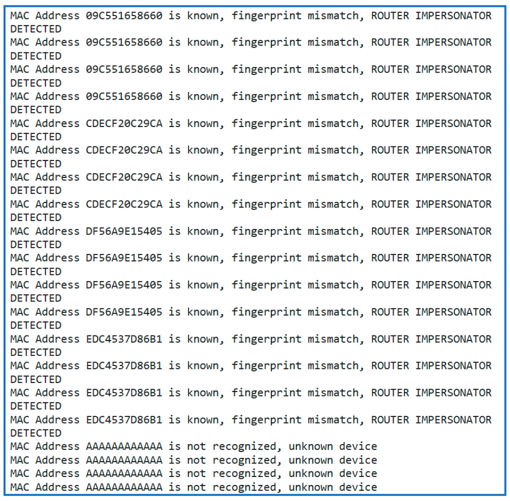
Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.0001

Figure 11. The training progress shows the accuracy and loss along with the iterations.

According to the results of the training progress graphic, the network converges to around 100% accuracy within the first epoch.

For the test frames, we obtained the predicted classes and calculated the test accuracy. The plotting of the confusion matrix for test data is shown in Figure 12. As was previously noted, using the synthetic dataset, complete classification accuracy was attained.



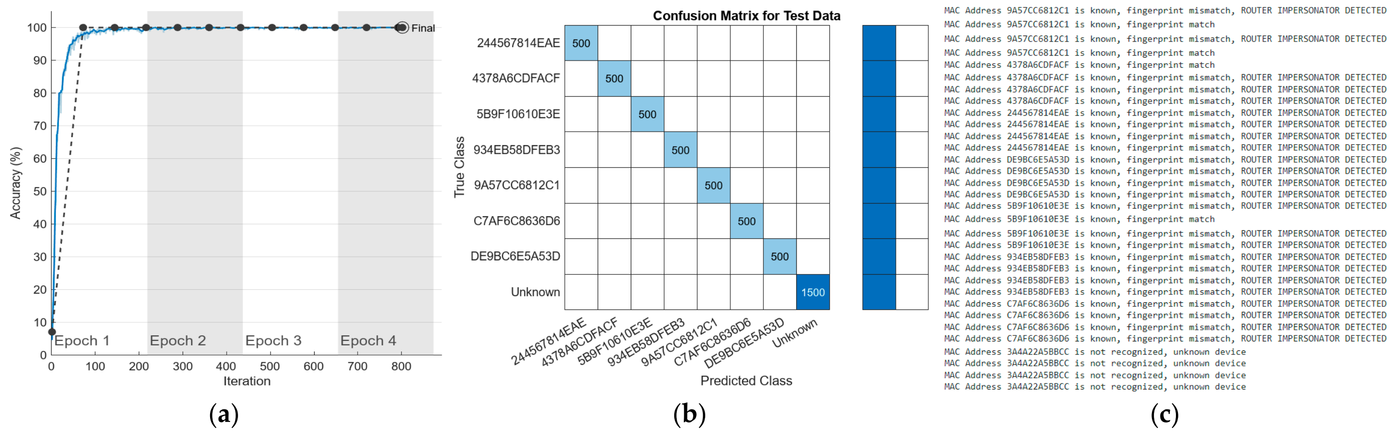


```
MAC Address 09C551658660 is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address 09C551658660 is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address 09C551658660 is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address 09C551658660 is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address CDECF20C29CA is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address CDECF20C29CA is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address CDECF20C29CA is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address CDECF20C29CA is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address DF56A9E15405 is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address DF56A9E15405 is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address DF56A9E15405 is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address DF56A9E15405 is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address EDC4537D86B1 is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address EDC4537D86B1 is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address EDC4537D86B1 is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address EDC4537D86B1 is known, fingerprint mismatch, ROUTER IMPERSONATOR
DETECTED
MAC Address AAAAAAAAAAAA is not recognized, unknown device
MAC Address AAAAAAAAAAAA is not recognized, unknown device
MAC Address AAAAAAAAAAAA is not recognized, unknown device
MAC Address AAAAAAAAAAAA is not recognized, unknown device
```

Figure 12. Plotting the confusion matrix for test data and a sample of the analytical results.

3.2. Scenario 2, Generating Random MAC Address for the Unknown Devices and 7 Known Routers

The operation of MAC address generation after applying channel impairments results in data frames for the router of the WLAN, a photo of the generated data is shown in Figure 13.



```
MAC Address 9A57CC6812C1 is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 9A57CC6812C1 is known, fingerprint match
MAC Address 9A57CC6812C1 is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 9A57CC6812C1 is known, fingerprint match
MAC Address 4378A6CDFACF is known, fingerprint match
MAC Address 4378A6CDFACF is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 4378A6CDFACF is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 4378A6CDFACF is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 244567814EAE is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 244567814EAE is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 244567814EAE is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 244567814EAE is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address DE9BC6E5A53D is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address DE9BC6E5A53D is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address DE9BC6E5A53D is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address DE9BC6E5A53D is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 5B9F10610E3E is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 5B9F10610E3E is known, fingerprint match
MAC Address 5B9F10610E3E is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 5B9F10610E3E is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 934EB58DFEB3 is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 934EB58DFEB3 is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 934EB58DFEB3 is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 934EB58DFEB3 is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address C7AF6C8636D6 is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address C7AF6C8636D6 is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address C7AF6C8636D6 is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address C7AF6C8636D6 is known, fingerprint mismatch, ROUTER IMPERSONATOR DETECTED
MAC Address 3A4A22A5BBCC is not recognized, unknown device
MAC Address 3A4A22A5BBCC is not recognized, unknown device
MAC Address 3A4A22A5BBCC is not recognized, unknown device
MAC Address 3A4A22A5BBCC is not recognized, unknown device
```

Figure 13. Scenario 2 (**a**) the training progress, (**b**) the confusion matrix for the tested data, and (**c**) router intrusion detections.

There are also four normally generated MAC addresses for the known routers and 10 different other MAC addresses for the unknown routers. The training options have been configured under the same conditions as scenario 1. According to the training progress of this scenario, the developed CNN also achieved 100% accuracy within the first epoch.

3.3. Network Evaluation

The performance of the developed architecture has been tested, which is represented by the obtained accuracy along with the number of devices used in the network as shown in Figure 14.

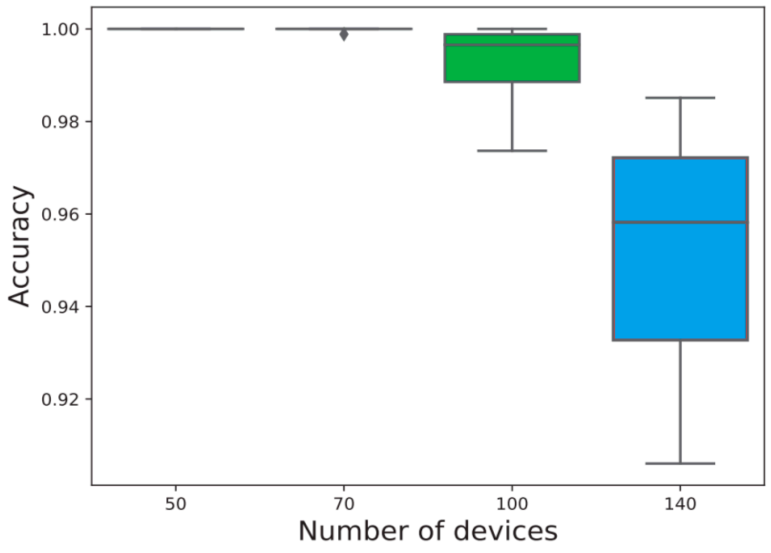


Figure 14. Accuracy versus number of devices for the developed architecture.

4. Discussion

According to this result, the performance of the proposed model is highly affected by the number of devices on the network. The main issue of applying this approach, in terms of scalability, is that it is required to re-train the whole CNN when any additional device is presented in the network.

Generalizability in deep learning is a critical concept to overcome the problem of overfitting. Overfitting occurs when a model learns the training data too well and performs poorly on unseen data because it has essentially memorized the training set. Therefore, to improve generalization and mitigate overfitting in this study, we implemented the following strategies:

- More data has been generated, where a 5000 Non-HT dataset of WLAN beacon frames for each router in the network. More data allows the model to learn a broader range of patterns and relationships.
- Monitor for early stopping, where monitoring the model's performance on a validation set during training and stopping when the performance starts to degrade. This prevents the model from continuing to overfit.
- Applying Cross-Validation to assess the model's performance on different subsets of the data. This can help to estimate how well the model will perform on unseen data.
- Simpler and appropriate feature set, where fewer features are less prone to overfitting, containing 6 parameters chosen to simplify the input representation while maintaining appropriate architecture, such as the L-LTF sequence frame Length, WLAN channel number, channel Band (Ghz), SNR (dB), and number of Known Routers and Unknown Routers.
- Ensemble Learning, where the combination of the predictions of multiple models was used to reduce overfitting helped to improve generalization as they can capture patterns that may be missed by a single model especially when dealing with complex I and Q datasets. The developed neural network architecture combines two (models) convolutional layers followed by three fully connected layers for processing complex data with both in-phase (I) and quadrature (Q) components, where the first layer learns the features independently in I and Q, and the next layer extracts the features combining I and Q together.
- Batch Normalization, the 7th layer of the developed architecture was Batch Normalization one, it normalizes the activations in intermediate layers during training, which can improve generalization and make the training process more stable.
- Using the Dropout layer in the 12th layer, where random neurons are "dropped out" during each training step. This prevents the network from relying too heavily on any one neuron.

Choosing the proper group of features is a challenging task that can cause scalability troubles when an extensive number of instruments are present, guiding to extended computational complexity during training. Furthermore, the inexperienced process of presenting arbitrary mixtures of impairments before training the CNN has three concerns:

(1) Communication effect: Naturally, adding impairments raises the BER. Therefore, to limit any negative effects on BER, controlled and cautious addition is required.
(2) Accuracy: Demodulated samples from two separate transmitters that were previously simple to distinguish may now seem clustered together due to changes made to where they were placed on the IQ plane. The classifier's performance can suffer as a result.
(3) Scalability: If an additional transceiver is presented in the network, then it is required to re-train the whole CNN, which is a computation- and time-heavy operation.

5. Conclusions and Future Investigation

This work presented a CNN-based fingerprinting RF model with computer-generated data and beacon frames to detect WLAN router impersonators. The developed deep CNN enhanced by ADAM optimizer is used to detect router impersonations of a WLAN via comparing the received signals, which includes a pair of MAC address/RF signature data for known and unknown devices. The results throughout the confusion matrix for the test data of the synthetic dataset demonstrated that the developed network architecture accurately classified the four generated MAC addresses of the known routers and the MAC address for the remaining 10 unknown routers with 100% accuracy. This demonstrates that even in the presence of random noise, the distinctive patterns produced by the impairments can still be recognized.

The experiment parameters included 4 known Routers, 10 Unknown Routers, 20 dB of SNR, 153 WLAN channel number, 5 Ghz channel Band, and 160 frame L-LTF Length in samples. All these values affect the obtained results. The challenging issue was with introducing impairments, which worsens the quality of service and raises the bit error rate (BER). For radios operating at varied SNR levels, the deterioration of impairments also varies [40]. The less degradation we must apply to radios to achieve the required BER, the lower the SNR. Assuming the SNR readings at the receiver side are quasi-static for duration T, enabling an average of SNR levels within each such time frame; we explore how to tackle this problem in this section. Such difficulties might arise when dealing with real-world noisy or dynamic environments. This study demonstrated how setting parameters like the number of known routers, unknown routers, and SNR can increase classification accuracy for comparable devices, further enhancing to account for new attack strategies and vulnerabilities that might emerge in the future.

Future areas of improvements can be performed by modifying the network architecture by changing: (1) the number of convolutional layers, (2) the number of fully connected layers, and (3) the Convolutional layer parameters (padding, number of filters, and filter size). The modification also can be done by testing the model under different RF impairments and channels by adjusting the following: (1) RF impairments (dc Offset Range, freq Offset Range, and phase Noise Range variables), (2) Channel noise level (SNR input of "*awgn*" function), and (3) Multipath profile of Rayleigh channel (Average Path Gains and Path Delays features object).

Author Contributions: Conceptualization, O.I.D.B., S.M.J. and Y.M.A.K.; methodology, Y.M.A.K.; software, H.K.H. and S.M.J.; validation, A.H.S. and H.K.H.; formal analysis, O.I.D.B.; investigation, Y.M.A.K.; resources, O.I.D.B.; data curation, H.K.H. and S.M.J.; writing—original draft preparation, S.M.J. and Y.M.A.K.; writing—review and editing, H.K.H.; visualization, O.I.D.B.; supervision, A.H.S.; project administration, A.H.S.; funding acquisition, A.H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: We acknowledge all support given from the Department of Medical Technical Institute, Northern Technical University, Mosul, Iraq.

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

References

1. Kavianpour, A.; Anderson, M.C. An Overview of Wireless Network Security. In Proceedings of the 4th IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2017 and 3rd IEEE International Conference of Scalable and Smart Cloud, SSC 2017, New York, NY, USA, 26–28 June 2017. [CrossRef]
2. Savaş, S.; Topaloğlu, N.; Ciylan, B. Analysis of mobile communication signals with frequency analysis method. *Gazi Univ. J. Sci.* **2012**, *25*, 447–454.
3. Savaş, S.; Topaloğlu, N. Data analysis through social media according to the classified crime. *Turk. J. Electr. Eng. Comput. Sci.* **2019**, *27*, 407–420. [CrossRef]
4. Mavridis, I.P.; Androulakis, A.I.E.; Halkias, A.B.; Mylonas, P. Real-life paradigms of wireless network security attacks. In Proceedings of the 2011 Panhellenic Conference on Informatics, PCI 2011, Kastoria, Greece, 30 September–2 October 2011. [CrossRef]
5. Meyer, U.; Wetzel, S. A man-in-the-middle attack on UMTS. In Proceedings of the 2004 ACM Workshop on Wireless Security, WiSe, Philadelphia, PA, USA, 1 October 2004. [CrossRef]

6. Bratus, S.; Cornelius, C.; Kotz, D.; Peebles, D. Active behavioral fingerprinting of wireless devices. In Proceedings of the WiSec'08: 1st ACM Conference on Wireless Network Security, Alexandria, VA, USA, 31 March–2 April 2008. [CrossRef]
7. Brik, V.; Banerjee, S.; Gruteser, M.; Oh, S. Wireless device identification with radiometric signatures. In Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM, San Francisco, CA, USA, 14–19 September 2008. [CrossRef]
8. Neumann, C.; Heen, O.; Onno, S. An empirical study of passive 802.11 device fingerprinting. In Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW 2012, Macau, China, 18–21 June 2012. [CrossRef]
9. Baza, M.; Nabil, M.; Mahmoud, M.M.E.A.; Bewermeier, N.; Fidan, K.; Alasmary, W.; Abdallah, M. Detecting Sybil Attacks Using Proofs of Work and Location in VANETs. *IEEE Trans. Depend. Secur. Comput.* **2020**, *19*, 39–53. [CrossRef]
10. Yang, J.; Chen, Y.; Trappe, W. Detecting sybil attacks in wireless and sensor networks using cluster analysis. In Proceedings of the 2008 5th IEEE International Conference on Mobile Ad-Hoc and Sensor Systems, MASS 2008, Atlanta, GA, USA, 29 September–2 October 2008. [CrossRef]
11. Xu, Q.; Zheng, R.; Saad, W.; Han, Z. Device fingerprinting in wireless networks: Challenges and opportunities. *IEEE Commun. Surv. Tutorials* **2015**, *18*, 94–104. [CrossRef]
12. Modi, C.; Patel, D.; Borisaniya, B.; Patel, H.; Patel, A.; Rajarajan, M. A survey of intrusion detection techniques in Cloud. *J. Netw. Comput. Appl.* **2013**, *36*, 42–57. [CrossRef]
13. Wu, Y.; Wei, D.; Feng, J. Network attacks detection methods based on deep learning techniques: A survey. *Secur. Commun. Netw.* **2020**, *2020*, 8872923. [CrossRef]
14. Huang, Y.; Zheng, H. Radio frequency fingerprinting based on the constellation errors. In Proceedings of the APCC 2012—18th Asia-Pacific Conference on Communications: "Green and Smart Communications for IT Innovation", Jeju, Republic of Korea, 15–17 October 2012. [CrossRef]
15. Candore, A.; Kocabas, O.; Koushanfar, F. Robust stable radiometric fingerprinting for wireless devices. In Proceedings of the 2009 IEEE International Workshop on Hardware-Oriented Security and Trust, HOST 2009, San Francisco, CA, USA, 27 July 2009. [CrossRef]
16. Niu, Y.; Xiong, W.; Li, Z.; Dong, B.; Fu, X. On the Identification Accuracy of the I/Q Imbalance Based Specific Emitter Identification. *IEEE Access* **2023**, *11*, 75462–75473. [CrossRef]
17. Radhakrishnan, S.V.; Uluagac, A.S.; Beyah, R. GTID: A Technique for Physical Device and Device Type Fingerprinting. *IEEE Trans. Depend. Secur. Comput.* **2015**, *12*, 519–532. [CrossRef]
18. O'Shea, T.J.; Corgan, J.; Clancy, T.C. Convolutional radio modulation recognition networks. In *Communications in Computer and Information Science*; Springer International Publishing: Cham, Switzerland, 2016. [CrossRef]
19. O'Shea, T.J.; Hoydis, J. *An Introduction to Machine Learning Communications Systems. Machine Learning Methods for Ecological Applications*; Springer International Publishing: Cham, Switzerland, 2017. [CrossRef]
20. Nguyen, N.T.; Zheng, G.; Han, Z.; Zheng, R. Device fingerprinting to enhance wireless security using nonparametric Bayesian method. In Proceedings of the IEEE INFOCOM, Shanghai, China, 10–15 April 2011. [CrossRef]
21. Tang, Y.; Wan, J.; Huang, B.; Lan, T. Improved dependent component analysis for hyperspectral unmixing with spatial correlations. In Proceedings of the International Symposium on Optoelectronic Technology and Application 2014: Image Processing and Pattern Recognition, Beijing, China, 13–15 May 2014. [CrossRef]
22. Franklin, J.; McCoy, D.; Tabriz, P.; Neagoe, V.; van Randwyk, J.; Sicker, D. Passive data link layer 802.11 wireless device driver fingerprinting. In Proceedings of the 15th USENIX Security Symposium, Vancouver, BC, Canada, 31 July 31–4 August 2006.
23. Duan, Y.; Lam, K.Y.; Lee, V.C.S.; Nie, W.; Liu, K.; Li, H.; Xue, C.J. Data Rate Fingerprinting: A WLAN-Based Indoor Positioning Technique for Passive Localization. *IEEE Sens. J.* **2019**, *19*, 6517–6529. [CrossRef]
24. Mazor, G.; Weizman, L.; Tal, A.; Eldar, Y.C. Low-rank magnetic resonance fingerprinting. *Med. Phys.* **2018**, *45*, 4066–4084. [CrossRef]
25. Halder, S.; Newe, T. Radio fingerprinting for anomaly detection using federated learning in LoRa-enabled Industrial Internet of Things. *Futur. Gener. Comput. Syst.* **2023**, *143*, 322–336. [CrossRef]
26. Zhang, P.; Shi, X.; Khan, S.U.; Ferreira, B.; Portela, B.; Oliveira, T.; Borges, G.; Domingos, H.; Leitão, J.; Mohottige, I.P.; et al. IEEE Draft Standard for Spectrum Characterization and Occupancy Sensing. *IEEE Access* **2019**, *9*.
27. Sankhe, K.; Belgiovine, M.; Zhou, F.; Angioloni, L.; Restuccia, F.; D'Oro, S.; Melodia, T.; Ioannidis, S.; Chowdhury, K. No Radio Left Behind: Radio Fingerprinting Through Deep Learning of Physical-Layer Hardware Impairments. *IEEE Trans. Cogn. Commun. Netw.* **2019**, *6*, 165–178. [CrossRef]
28. Model-Based Machine Learning for Communications. In *Machine Learning and Wireless Communications*; Cambridge University Press: Cambridge, UK, 2022. [CrossRef]
29. Hamza, A.H.; Hussein, S.A.; Ismaeel, G.A.; Abbas, S.Q.; AbdulZahra, M.M.; Sabry, A.H. Developing Three Dimensional Localization System Using Deep Learning and Pre-Trained Architectures for Ieee 802.11 Wi-Fi. *East.-Eur. J. Enterp. Technol.* **2022**, *4*, 41–47. [CrossRef]
30. Al-Shoukry, S.; Jawad, B.J.M.; Musa, Z.; Sabry, A.H. Development of predictive modeling and deep learning classification of taxi trip tolls. *East.-Eur. J. Enterp. Technol.* **2022**, *3*, 6–12. [CrossRef]

31. Ojo, M.O.; Zahid, A. Deep Learning in Controlled Environment Agriculture: A Review of Recent Advancements, Challenges and Prospects. *Sensors* **2022**, *22*, 7965. [CrossRef]
32. Zhou, K.; Wang, W.; Hu, T.; Deng, K. Time series forecasting and classification models based on recurrent with attention mechanism and generative adversarial networks. *Sensors* **2020**, *20*, 7211. [CrossRef]
33. Wang, Y.; Xiao, Z.; Cao, G. A convolutional neural network method based on Adam optimizer with power-exponential learning rate for bearing fault diagnosis. *J. Vibroeng.* **2022**, *24*, 666–678. [CrossRef]
34. Elshamy, R.; Abu-Elnasr, O.; Elhoseny, M.; Elmougy, S. Improving the efficiency of RMSProp optimizer by utilizing Nestrove in deep learning. *Sci. Rep.* **2023**, *13*, 8814. [CrossRef]
35. Lydia, A.A.; Francis, F.S. Adagrad-An Optimizer for Stochastic Gradient Descent. *Int. J. Inf. Comput. Sci.* **2019**, *6*, 566–568.
36. Yadav, R.K.; Anubhav. PSO-GA based hybrid with Adam Optimization for ANN training with application in Medical Diagnosis. *Cogn. Syst. Res.* **2020**, *64*, 191–199. [CrossRef]
37. Sankhe, K.; Belgiovine, M.; Zhou, F.; Riyaz, S.; Ioannidis, S.; Chowdhury, K. ORACLE: Optimized Radio clAssification through Convolutional neuraL nEtworks. In Proceedings of the IEEE INFOCOM, Paris, France, 9 April–2 May 2019. [CrossRef]
38. *IEEE Std 802.11-2007*; Part 11: Wireless Lan Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Computer Society: Washington, DC, USA, 2007.
39. Skordoulis, D.; Ni, Q.; Chen, H.H.; Stephens, A.P.; Liu, C.; Jamalipour, A. IEEE 802.11N MAC frame aggregation mechanisms for next-generation high-throughput WLANs. *IEEE Wirel. Commun.* **2008**, *15*, 40–47. [CrossRef]
40. Wang, S.; Chen, Y.; Leeson, M.; Beaulieu, N.C. Channel capacity and bit error rate optimization of the ultra-wide bandwidth transmitted-reference receiver. *Wirel. Commun. Mob. Comput.* **2013**, *13*, 1657–1670. [CrossRef]