Deep Learning for RF Device Fingerprinting in Cognitive Communication Networks

Kevin Merchant, Shauna Revay, George Stantchev, and Bryan Nousain

***Abstract*—With the increasing presence of cognitive radio networks as a means to address limited spectral resources, improved wireless security has become a necessity. In particular, the potential of a node to impersonate a licensed user demonstrates the need for techniques to authenticate a radio's true identity. In this paper, we use deep learning to detect physical-layer attributes for the identification of cognitive radio devices, and demonstrate the performance of our method on a set of IEEE 802.15.4 devices. Our method is based on the empirical principle that manufacturing variability among wireless transmitters that conform to the same standard creates unique, repeatable signatures in each transmission, which can then be used as a fingerprint for device identification and verification. We develop a framework for training a convolutional neural network using the time-domain complex baseband error signal and demonstrate 92.29% identification accuracy on a set of 7 2.4 GHz commercial ZigBee devices. We also demonstrate the robustness of our method over a wide range of signal-to-noise ratios.**

***Index Terms*—Communication system security, neural networks, ZigBee, cognitive radio, personal area networks, radio communication, pattern recognition, classification algorithms.**

I. Introduction

COGNITIVE Radio Networks (CRNs) are becoming progressively more useful due to the expanding presence of wireless technology and the increasingly crowded spectrum [1]. Cognitive radio is a model in which nodes adapt their transceiver parameters according to current spectrum usage in order to maximize spectral efficiency. In this model, *primary users* (PUs) with known channel assignments are given the highest priority for transmission, though they only occupy their frequency bands for finite periods of time. *Secondary users* (SUs) perform spectrum sensing and fill in temporarily unoccupied frequencies so that they do not interfere with the operation of PUs. To this end, SUs constantly monitor a set of frequency bands, sensing the presence of PUs and searching for unoccupied channels [2].

The ability to detect the presence of a PU is paramount to the functionality of CRNs. However, this mechanism introduces vulnerabilities that may allow an attacker to disguise themselves as a PU that occupies a licensed part of the spectrum and cause a denial-of-service (DoS) attack for SUs. This manner of attack is referred to as Primary User Emulation (PUE) [3] and it highlights the need for improved measures in CRNs that verify the identity of transmitters. Radio Frequency (RF) fingerprinting methods have been proposed as a countermeasure against PUE attacks [4]–[6].

The authors are with the US Naval Research Laboratory, Washington, DC 20375 USA (e-mail: kevin.merchant@nrl.navy.mil)

It has been demonstrated that cognitive radios allow for reduced intermodulation in the transmitted signal by opportunistically selecting frequency bands to minimize out-of-band emissions [7]. In particular, this permits the use of low-cost devices with nonlinear front-end components in CRNs. For that reason, novel RF fingerprinting algorithms are becoming increasingly applicable in CRNs composed of low-cost devices with consumer-grade front-ends.

RF fingerprinting aims to identify transmitters by characterizing device-specific features present in their emitted analog signals. These features are primarily a result of hardware variability in the device's analog transmitter components. To this end we have studied RF fingerprinting based on the application of deep learning classification methods and we demonstrate its feasibility on a collection of low-cost devices, typically used in Wireless Personal Area Networks (WPANs). These devices implement the IEEE 802.15.4-based ZigBee standard, and are increasingly common in application areas such as home and building automation, health care management, retail, and smart grid technologies [8]. Typically, these networks are secured via cryptographic keys and thus are only as secure as the keys themselves [9], [10]. Previous work has demonstrated key-extraction attacks on ZigBee both via physical access to the device and over the air [11], [12], emphasizing the need for authentication based on physical-layer attributes which are not easily replicated. In particular, methods are needed which can distinguish a transmitter from all others, even those of the same model and the same manufacturer, without relying on bit-level information such as MAC addresses or cryptographic keys. Security of low cost, short-range wireless devices will be vital as the internet of things (IoT) continues to expand.

To address the limits of key-based security, a number of previous studies have demonstrated physical-layer fingerprinting techniques by developing a feature extractor appropriate for the target class of signals and using machine learning algorithms to develop a classifier. However, previous studies have largely been limited to classifiers based on hand-crafted features that require significant knowledge of the types and characteristics of signals present. Motivated by the success of deep learning methods at complex classification tasks in image and speech recognition, we apply a convolutional neural network (CNN) to the RF fingerprinting problem. Our network operates on the time-domain complex baseband error signal and unlike many previous approaches which are limited to operating only on the signal preamble, our method can utilize the full transmission for training and evaluation, regardless of the signal payload.

This paper is organized as follows. Section II gives a brief

introduction to the IEEE 802.15.4 standard and discusses previously published methods for RF fingerprinting as well as several prior applications of deep learning to RF signals. Section III introduces our CNN approach and the necessary signal pre-processing steps. Section IV presents our identification and verification results for a set of 7 commercial off-the-shelf ZigBee devices in both outdoor and lab settings. Section V provides concluding remarks and future research directions.

II. Background

A. IEEE 802.15.4

The IEEE 802.15.4 specification [9] defines the physical and media access control (MAC) standards for ZigBee devices. While the devices used in our experiments implement the full ZigBee Pro stack, ZigBee is a higher level protocol built on top of the physical layer specifications defined in the IEEE 802.15.4 standard [9], [10]. The standard requires offset quadrature phase-shift keying (O-QPSK) with half-sine pulse shaping and direct-sequence spread spectrum (DSSS) modulation at a data rate of 250 kbps. The half-sine pulse shaping used allows the data to be demodulated noncoherently using a binary FSK demodulator. Because of the DSSS technique used, 32 symbols are transmitted for every data byte. The first 5 bytes of any IEEE 802.15.4 O-QPSK transmission include a 4 byte preamble of zeros, followed by the 1 byte start frame delimiter (SFD) 0xA7. These 5 bytes are collectively known as the synchronization header (SHR) field of the packet.

B. Related Work

A commonly used feature extraction technique for WPAN devices such as ZigBee and Z-Wave is to divide a fixed header portion of the signal, such as the preamble or SHR, into a number of equal-sized regions and compute statistics such as variance, skewness, and kurtosis on the instantaneous amplitude, phase, or frequency signal in each region [13]–[16]. Classification algorithms can then learn a distribution of feature vectors from training data in order to distinguish transmissions from different devices. Previous studies have found that phase features are the most useful for distinguishing ZigBee devices [15].

Another common approach to RF fingerprinting involves measuring the transient behavior of the device. A probabilistic neural network has previously been trained to distinguish 8 IEEE 802.11b WiFi cards using the transient amplitude signal of WiFi transmissions [17]. Our approach is different in that it learns features from the steady state portion of the transmission. This is useful as the steady state portion is typically received at a higher average power level and has a much longer duration than the transient. Alternatively, methods based on Hilbert-Huang [18] and wavelet [19] transforms have been demonstrated for successful RF fingerprinting.

Neural networks have been used to develop models of nonlinear power amplifiers and perform predistortion [20], [21]. In a similar manner, our method develops a model of the signal imperfections demonstrated in each device. Rather than using this model at the transmitter to compensate the errors and improve the transmitted signal, our method detects the errors after transmission and uses them for classification. In addition, a simple multilayer perceptron (MLP) network operating on cross-correlation values has been used to identify the presence or absence of a receiver based on its unintended emissions [22]. Neural networks have also been used to identify individual radar transmitters, though the network input parameters, such as pulse width and pulse repetition interval, were hand-selected for radar applications [23], and would not apply to a typical communications setting.

Recently, deep learning has been applied to a number of wireless communications problems. A CNN operating on the time-domain complex baseband signal has demonstrated high performance at classifying the modulation scheme used by a transmitter [24]. In addition, a CNN has been used for wireless interference identification for IEEE 802.11, 802.15.4, and 802.15.1 devices in order to classify wireless devices by allocated frequency channel and the type of wireless technology [25].

Our research is novel in the following sense. We use time-domain complex baseband error signals to train a CNN which we then apply to a test signal to verify if it came from a specific device with a high degree of accuracy. We are also able to identify the source device without a particular target device to compare it against. While previous methods have achieved success at the same goal, they typically utilize a fixed preamble signal [13]–[16], [26] or transients [17], [27] of a transmission for classification. Our method is unrestricted by which part of the signal is used, and does not assume a common message is being transmitted. This is a clear benefit in the case of signals without preambles or with short-lived transient behavior. Our method operates directly on the error signal and does not require manually selecting relevant features from it. This allows the network to learn the features that best distinguish the devices without requiring any *a priori* knowledge of the sorts of features appropriate for the target devices.

Furthermore, we apply a frequency correction to each transmission to remove any device-dependent carrier frequency offset that may be present. To the best of our knowledge, frequency correction was not performed in previous RF fingerprinting experiments. Frequency correction makes the classification problem significantly more challenging, as small carrier frequency offsets persist at low signal-to-noise ratio. However, by removing this frequency offset we remove a feature that could easily be spoofed by a malicious transmitter with a precise oscillator. By performing this correction, we ensure our neural network learns non-trivial intrinsic characteristics of the devices. This gives promise to the scalability and the robustness of our results since we have factored out features that may not be distinct in the presence of a large number of devices, and features that are easy to spoof.

In addition, demonstrating the performance of our technique on frequency-corrected data makes our technique particularly relevant in a CRN context, where frequency bands are used opportunistically and an exact fixed carrier frequency is rarely known *a priori*. By removing frequency offsets and bringing each transmission as close as possible to baseband, we remove one source of possible variation across frequency bands,

increasing the likelihood that our method will be successful in a CRN context.

III. Methods

A. Problem Statement

We define two classification problems. *Identification* seeks to select which device among those in the training set produced a given transmission. This is a multi-class problem and performance may be evaluated using a confusion matrix and correct classification rates. For identification, we develop a single CNN for each experiment to assign transmissions to devices in the training set.

Verification seeks to confirm whether or not a given transmission originates from a particular device. In a practical setting, a device may be verified against the device model whose source address matches the one claimed in the transmission. This is a binary one-vs-all classification problem, and performance may be evaluated using receiver operating characteristics (ROC) and precision-recall curves. For verification, one CNN is trained per device for each experiment.

B. Data Collection

We demonstrate our technique on a collection of 7 Digi XBP24CZ7SITB003 ZigBee Pro devices operating in the 2.4 GHz band. The device under test (DUT) was added to a network with another device, known as the coordinator in the ZigBee protocol, to allow transmission. A unique set of packets with random 32 byte payloads was generated in MATLAB for each device and sent to the DUT using the Digi XCTU software [28]. All devices transmitted on IEEE 802.15.4 channel 11, corresponding to a 2.405 GHz center frequency, and were configured to transmit at their maximum power level of 18 dBm. For each device, approximately 28 seconds of data was digitized at a sampling rate of 16 MHz, and the received signal was downconverted to baseband and recorded on a Rohde & Schwarz FSW67 signal and spectrum analyzer with external SRS FS725 reference.

We demonstrate our technique via two experiments: In one experiment, a CNN is trained on data collected outdoors through a noisy channel. In a second experiment, a CNN is trained on data collected in a lab environment with simulated additive white Gaussian noise. For the outdoor experiment, both the coordinator and the DUT were connected to Pulse Electronics W1030 omni-directional antennas and DUTs were located approximately 30 meters from the receiver antenna in an urban outdoor environment where the estimated in-band signal-to-noise ratio of the received transmissions is approximately 28 dB. For the lab experiment, DUTs were hardwired to a power splitter which was connected directly to the signal and spectrum analyzer, and the transmissions are assumed to be virtually noise-free.

C. Data Pre-processing

Prior to training the network, a number of pre-processing steps are taken in MATLAB. First, each individual transmission is extracted from the recording. Next, the estimated ideal signal is used to synchronize the transmissions in phase, frequency, and time. Because the signals are oversampled, each signal is then low-pass filtered to remove components that fall outside the transmitter passband. Finally, the error signal for each transmission is calculated and saved to its own file. The error signal for each transmission is used as an input to the CNN.

1) Extraction: Approximate start and stop times for each transmission were detected in MATLAB by thresholding the amplitude differences between adjacent samples. Each transmission was aligned to a reference signal containing the 5 byte SHR by cross-correlating the instantaneous frequency of the received and reference signals. The final alignment selected the time lag between signals that maximized the cross-correlation and produced a valid checksum of the demodulated data. Transmissions from the coordinator were discarded, and to demonstrate that the method is unaffected by the data contents of the transmission, only the portion of the signal containing the 32 byte random payload and the 2 byte checksum from each transmission was retained. Each remaining signal was scaled by dividing all samples by the RMS of the magnitude signal so that slight power level differences between transmitters did not permit trivial classification. Each of the first 1,000 valid transmissions from each device for each experiment was processed in this manner and saved to its own MATLAB file.

2) Additive Noise: For the lab data experiment, each transmission was included in the dataset several times with varying levels of simulated additive white Gaussian noise. After extraction, additive noise was imposed on each transmission via MATLAB's *awgn()* function. Each transmission was included in the dataset 8 times for each SNR in the range $\{10, 15, 20, \ldots, 40\}$ dB. Each instance of the same transmission at a particular SNR uses a different random noise signal. By including each transmission several times, we are able to increase the amount of training data given a fixed amount of lab data. The network was trained on the entire training dataset at once rather than training at a single SNR.

3) Synchronization: A common problem in demodulation of wireless transmissions is that of synchronization between the transmitter and receiver. In this case, the signals may be demodulated noncoherently. However, because we are using the baseband error signal, we perform synchronization on each transmission individually to remove fixed frequency and phase offsets from the signal and improve the sample timing between the transmitter and receiver. In doing so, we aim to remove trivial features that fluctuate randomly or are easily spoofed, and learn more reliable features for classification. Further, this reduces sources of bias in the experiment since the phase relationship between the transmitter and receiver oscillators could change by simply resetting either device.

Since the transmissions can be demodulated noncoherently and we are targeting a known modulation scheme, we assume that the ideal discrete-time baseband signal can be estimated based on the demodulated symbols. We model the synchronization error between the transmitter and receiver as a fixed frequency and phase offset which are constant for the entirety of a single transmission, but which may be different for each transmission. For simplicity, we model these as offsets in

the transmitter from the receiver's ideal values. That is, for complex baseband signal $x_{\text{ideal}}(t)$, time index t, and assuming no other channel distortion, the ideal passband signal is given by:

$$y_{\text{ideal}}(t) = \text{Real}(x_{\text{ideal}}(t)e^{j\omega_c t}),$$

where ω_c is the ideal carrier frequency. Due to the presence of the phase and frequency offset, the actual transmitted signal is given by:

$$y_{\text{sent}}(t) = \text{Real}(x_{\text{ideal}}(t)e^{j((\omega_c+\omega_o)t+\theta_o)}),$$

where ω_o and θ_o are the transmitter's frequency and phase offset, respectively. After downconversion to baseband at the receiver, the received signal is:

$$x_{\text{meas}}(t) = x_{\text{ideal}}(t)e^{j(\omega_o t+\theta_o)}.$$

To perform carrier synchronization, our goal is to find frequency correction ω_f and phase correction θ_f such that:

$$x_{\text{meas}}(t)e^{j(\omega_f t+\theta_f)} \approx x_{\text{ideal}}(t).$$

If we model $x_{ideal}(t)$ as an arbitrary time-dependent signal in the complex domain, such that $x_{\text{ideal}}(t) = A(t)e^{j\theta(t)}$, then this becomes:

$$A(t)e^{j(\theta(t)+\omega_o t+\theta_o+\omega_f t+\theta_f)} \approx A(t)e^{j(\theta(t))}.$$

If we define $\angle x_{\text{ideal}}(t)$ and $\angle x_{\text{meas}}(t)$ to be the phase angle of the ideal and measured baseband signals, respectively, then the following two expressions are equivalent:

$$-\omega_f t - \theta_f \approx (\theta(t) + \omega_o t + \theta_o) - \theta(t),$$

$$\omega_f t + \theta_f \approx \text{unwrap}(\angle x_{\text{ideal}}(t) - \angle x_{\text{meas}}(t)).$$

We use linear regression to find ω_f and θ_f that minimize the following expression:

$$\underset{\omega_f,\theta_f}{\text{argmin}} \sum_t \left(\omega_f t + \theta_f - \text{unwrap}(\angle x_{\text{ideal}}(t) - \angle x_{\text{meas}}(t))\right)^2.$$

Synchronization is typically performed with phase-locked loops to recover the carrier prior to demodulation. In this case, since the signal may be demodulated noncoherently and the estimated ideal signal may be generated, linear regression is used to find optimal values of ω_f and θ_f and to perform the frequency and phase corrections for each transmission.

Finally, the sample timing of each transmission is synchronized to the estimated ideal signal. To start, the estimated ideal signal is generated at a sample rate of 160 MHz and the received signal is interpolated by a factor of 10. The two signals are aligned by maximizing the cross-correlation over possible lag values in the range $[-9, 9]$. Rather than shifting the estimated ideal signal, the received signal is always shifted and, if necessary for alignment, the first 10 samples of the estimated ideal signal are removed. Both signals are then downsampled by a factor of 10 to restore the original 16 MHz sample frequency. By only shifting the received signal, the sample timing of the estimated ideal signal is unchanged and at most 1 sample is lost.

4) Error Signal Generation: Each signal was forward-backward filtered through a 4^{th} order Butterworth filter with a 2 MHz passband. The ideal signal was subtracted from each transmission resulting in the error signal for that transmission.

Any device-dependent features that may be useful for classification will not be present in the ideal signal. By subtracting the estimated ideal signal from the recorded transmissions to generate the error signal, we allow the network to ignore the portion of the signal that would be common across devices and focus on the differences which may be caused by features specific to each device, such as power amplifier nonlinearities [29] or oscillator imperfections [30].

5) Dataset Generation: Each signal was split into N non-overlapping segments of M consecutive samples. The network was trained on and learned to classify these segments of M samples, and the output of the network on each segment was combined later to classify each overall transmission. This splitting technique is useful for multiple reasons. For one, reducing the length of the network input signal decreases the size of the network, allowing for faster training and likely requiring less training data than a larger network. In addition, by splitting the signal into many smaller components, the method can be adapted for use with signals of arbitrary lengths by simply adjusting the threshold for classification by the number of M-sample segments in the signal.

The complex data from the time-series error signal was split into real and imaginary parts and treated as two separate input channels. Each time-series signal was normalized by subtracting off the mean value and dividing by the standard deviation, computed separately on both the real and imaginary parts for a given transmission. One dataset was created for the outdoor experiment, and a second for the indoor experiment. Each full dataset of 7,000 transmissions was randomly partitioned into 80% training data, 10% validation data, and 10% testing data with all N segments of a given transmission belonging to the same set. The same set assignment was used for both the identification and verification problems for a given dataset. The order of the segments was randomly shuffled for the training dataset, and then the resulting datasets were saved in HDF5 format.

D. Convolutional Neural Network

A deep CNN was used to solve the identification and verification problems, with the network structures given in Tables I, II, and III. The exponential linear unit (ELU) [31] was selected as the activation function for all applicable layers except the output layer where Softmax was used. The identification networks for the outdoor experiment and lab experiment have a total of 314,039 and 1,074,727 trainable parameters, respectively. The verification networks for both experiments have a total of 255,042 trainable parameters. Categorical cross-entropy was used to compute the loss.

For these experiments, there are $K = 7$ devices, $N = 17$ segments per transmission, and $M = 1,024$ samples per segment. Values for N and M were determined experimentally. Intuitively, $M = 1,024$ corresponds to segments of length 2 bytes. As demonstrated in our results, this choice

TABLE I
THE LAYERS OF THE IDENTIFICATION NETWORK FOR THE OUTDOOR DATA, ALONG WITH THE NUMBER OF PARAMETERS AND ACTIVATION FUNCTION OF EACH LAYER.

Layer	Dimension	Parameters	Activation
Input	1,024x2	-	-
Convolution 1D	128x19	4,992	ELU
Max Pooling	2	-	-
Convolution 1D	32x15	61,472	ELU
Max Pooling	2	-	-
Convolution 1D	16x11	5,648	ELU
Max Pooling	2	-	-
Flatten	-	-	-
Dense	128	239,744	ELU
Dropout (0.5)	-	-	-
Dense	16	2,064	ELU
Dropout (0.5)	-	-	-
Dense	7	119	Softmax

TABLE II
THE LAYERS OF THE IDENTIFICATION NETWORK FOR THE LAB DATA WITH ARTIFICIAL NOISE, ALONG WITH THE NUMBER OF PARAMETERS AND ACTIVATION FUNCTION OF EACH LAYER.

Layer	Dimension	Parameters	Activation
Input	1,024x2	-	-
Convolution 1D	128x19	4,992	ELU
Max Pooling	2	-	-
Convolution 1D	32x15	61,472	ELU
Max Pooling	2	-	-
Flatten	-	-	-
Dense	128	999,552	ELU
Dropout (0.5)	-	-	-
Dense	64	8,256	ELU
Dropout (0.5)	-	-	-
Dense	7	455	Softmax

TABLE III
THE LAYERS OF THE VERIFICATION NETWORKS, ALONG WITH THE NUMBER OF PARAMETERS AND ACTIVATION FUNCTION OF EACH LAYER.

Layer	Dimension	Parameters	Activation
Input	1,024x2	-	-
Convolution 1D	32x19	1,248	ELU
Max Pooling	2	-	-
Convolution 1D	128x19	77,952	ELU
Max Pooling	2	-	-
Convolution 1D	32x15	61,472	ELU
Max Pooling	2	-	-
Convolution 1D	16x11	5,648	ELU
Max Pooling	2	-	-
Flatten	-	-	-
Dense	128	106,624	ELU
Dropout (0.5)	-	-	-
Dense	16	2,064	ELU
Dropout (0.5)	-	-	-
Dense	2	34	Softmax

of segment length is long enough to detect repetitive patterns in the transmissions. However, this choice of M is also short enough to permit a relatively small input layer, and thus a CNN of a manageable size. Finally, M corresponding to a 2 byte segment allows for the scalability of the method to any signal with a length that is a multiple of 2 bytes. Therefore, at most 1 byte from an arbitrary transmission would be discarded for classification purposes.

For identification, a network was trained on all 7 devices, with each device representing a class. For verification, the network was trained on all 7 devices, with one device selected as the positive class for each model and the remaining devices all considered as a single negative class.

1) Classification: For the identification problem with K devices, the network output is a length-K vector interpreted as the set of estimated probabilities that the selected M-sample error signal belongs to each of the K devices. For verification, the network output is a length-2 vector, representing the estimated probability distribution of the event that the signal matches the device associated with the positive class for that network.

To assign a class label to the overall transmission x, we treat each of the M-sample vectors x_i for $i = 1, \ldots, N$ as mutually independent. Of course, this assumption is overly simplistic, as all M-sample vectors belong to the same transmission. Still, it provides a simple way to combine the estimated probability distributions of the different segments into a single metric and also provides strong classification performance. With this assumption, and denoting the estimated probability that segment x_i belongs to class ω_k as $p(\omega_k; x_i)$, the estimated probability that all segments x_i, $i = 1, \ldots, N$ belong to class ω_k is given by:

$$p(\omega_k; x_1, \ldots, x_N) = \prod_{i=1}^{N} p(\omega_k; x_i).$$

Since it is known a priori that all x_i for $i = 1, \ldots, N$ come from the same transmission x, and thus belong to the same class, the estimated probability $p(\omega_k; x)$ that transmission x belongs to class ω_k is given by:

$$p(\omega_k; x) = \frac{p(\omega_k; x_1, \ldots, x_N)}{\sum_{r=1}^{K} p(\omega_r; x_1, \ldots, x_N)}.$$

Transmission x may then be assigned to the class with the highest estimated probability. Because of numerical precision concerns when a large number of segments are used, we equivalently assign x to the class ω_k which maximizes:

$$\sum_{i=1}^{N} \log(p(\omega_k; x_i)). \tag{1}$$

2) Training: For the identification problem, the proportion of samples in each class was approximately $1/7$, with some variation due to the randomness of the split. Since verification is a one-vs-all problem, approximately $1/7$ of the available data belonged to the positive class, and $6/7$ of the data belonged to the negative class for each verification problem. During identification, all classes were given uniform weight. During verification, the positive class was given a weight of

TABLE IV
THE CONFUSION MATRIX FOR THE 7 DEVICE IDENTIFICATION PROBLEM FOR THE OUTDOOR DATA INDICATING THE NUMBER OF TRANSMISSIONS IN EACH CLASS OF THE TEST SET AND THEIR CLASSIFICATION.

Actual Device \ Predicted Device	1	2	3	4	5	6	7
1	85	2	1	0	0	0	1
2	0	93	2	0	0	0	9
3	1	8	92	0	0	0	0
4	0	3	0	103	0	0	0
5	0	0	2	0	86	14	2
6	1	0	0	0	3	87	0
7	0	0	0	5	0	0	100

6 and the negative class was given a weight of 1 due to the class imbalance.

Glorot uniform initialization [32] was used for kernel initialization of all convolutional and dense layers, and all bias vectors were initialized to the zero vector. The Adam optimizer and its original parameters was used for training [33]. The batch size was set to 1,024 segments and each batch was randomly shuffled at each epoch during training.

During training, only the classification performance of each 1,024-sample segment is considered. Each overall transmission is not classified during the training phase as the loss is computed for the segments only.

Validation was performed after every epoch. For the identification problem, a classification of each transmission in the validation set is performed during validation using Equation 1 as the confidence metric for each class. The classification accuracy on the validation set was tracked, and training was terminated when the accuracy did not improve for R consecutive evaluations on the validation set, at which point the best performing model was saved and used for testing. For the outdoor identification problem $R = 20$, and for the outdoor verification problem $R = 10$. For the identification problem with lab data $R = 10$, and the verification problem with lab data had a value of $R = 5$. For verification, each transmission is assigned to either the positive or negative class, and the ROC was calculated by thresholding the values calculated using Equation 1. The area under the curve (AUC) was calculated for the ROC, and the model with the highest AUC was tracked. Training was terminated when the AUC did not improve for R consecutive passes through the validation set, and the best performing model was saved.

3) Testing: During testing, the remaining 10% of transmissions were split into segments of 1,024 samples and given a forward pass through the network and the metric given in Equation 1 was calculated for each transmission. For identification, these values were compared, and the device with the maximum value was assigned as the class label. For verification, these values were thresholded to produce ROC and precision-recall curves for each verification network.

4) Implementation: A NVIDIA GeForce GTX 1080 Ti GPU was used to train and test the network. All data preprocessing was done in MATLAB R2017a and the CNN was implemented in Python using Keras 2.0.9 [34] with TensorFlow 1.3.0 [35] as the backend.

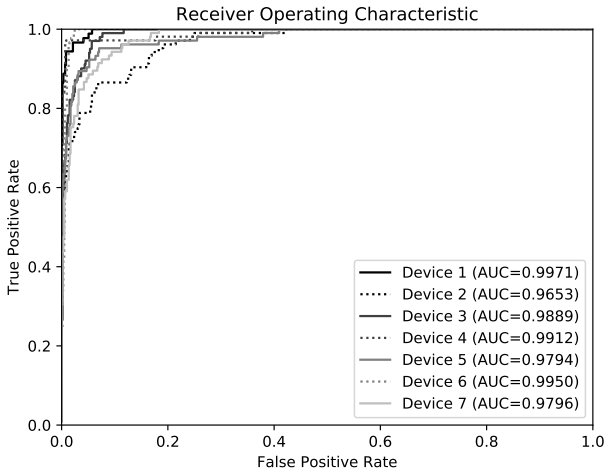

Fig. 1. Receiver operating characteristic curves for each of the 7 verification problems for the outdoor data. The area under the curve (AUC) is included in the legend for each device.

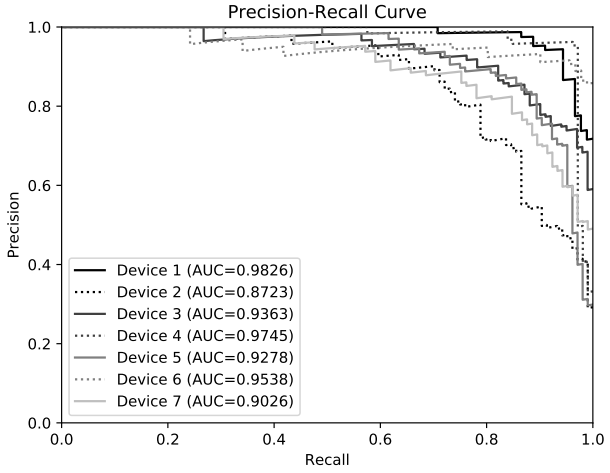

Fig. 2. Precision-recall curves for each for the 7 verification problems for the outdoor data, with the AUC for each curve given in the legend.

TABLE V
THE NUMBER OF EPOCHS UNTIL THE BEST MODEL WAS FOUND AND THE TOTAL TRAINING TIME FOR EACH NETWORK FOR THE OUTDOOR DATA. (NOTE THAT TRAINING CONTINUED AFTER THE BEST MODEL WAS OBTAINED AND THIS IS INCLUDED IN THE TOTAL TRAINING TIME.)

Network	Epochs	Training Time (minutes)
Identification	65	24.29
Verification 1	41	12.87
Verification 2	29	9.89
Verification 3	45	14.06
Verification 4	28	9.60
Verification 5	55	16.36
Verification 6	14	6.10
Verification 7	65	19.25

TABLE VI
THE NUMBER OF EPOCHS UNTIL THE BEST MODEL WAS FOUND AND THE TOTAL TRAINING TIME FOR EACH NETWORK FOR THE LAB DATA. (NOTE THAT TRAINING CONTINUED AFTER THE BEST MODEL WAS OBTAINED AND THIS IS INCLUDED IN THE TOTAL TRAINING TIME.)

Network	Epochs	Training Time (minutes)
Identification	17	482.14
Verification 1	1	87.68
Verification 2	3	114.03
Verification 3	1	87.30
Verification 4	3	118.82
Verification 5	2	100.19
Verification 6	1	89.52
Verification 7	1	87.89

TABLE VII
THE OVERALL ACCURACY AT EACH SNR FOR THE LAB EXPERIMENT IDENTIFICATION PROBLEM.

SNR (dB)	Accuracy (%)
10 dB	73.73
15 dB	85.45
20 dB	89.92
25 dB	90.61
30 dB	91.13
35 dB	91.38
40 dB	91.38

IV. RESULTS

Results from the outdoor and lab experiments are given in Figures 1-2 and Tables IV - VIII . For both experiments, overfitting was observed during the late stages of training. This is evident because early on, the classification accuracy of the CNN on the training set approximately matched that of the validation set, while near the end of training the validation accuracy peaked, and the training accuracy continued to improve. For this reason, we track the model with the best performance on the validation set, and end training after the validation performance metrics stop improving as defined in Section III.

A. Outdoor Experiment

A confusion matrix for the identification problem is shown in Table IV. The overall correct identification rate is 92.29% and the correct classification rate over all 1,024 sample segments in the test set is 77.01%. This demonstrates that the metric given in Equation 1 combines the results of the 1,024-sample segments in a way that boosts the classification performance over that of a single segment. Results for the verification problem are given in Figures 1 and 2. The mean ROC AUC of all devices is 0.9852 indicating very strong verification performance. Additional information regarding the amount of training for each network is given in Table V.

TABLE VIII
THE AVERAGE ROC AUC AND AVERAGE PRECISION-RECALL AUC AT EACH SNR FOR THE LAB EXPERIMENT VERIFICATION PROBLEM.

SNR (dB)	ROC AUC	Precision-Recall AUC
10 dB	0.9270	0.6935
15 dB	0.9594	0.8213
20 dB	0.9716	0.8820
25 dB	0.9757	0.9016
30 dB	0.9770	0.9070
35 dB	0.9774	0.9092
40 dB	0.9775	0.9103

B. Lab Experiment

Table VI describes the number of epochs of training until the best network parameters were found, as measured by the performance on the validation set, as well as the total training time in minutes for each network. It is important to note that since the network and training dataset in the lab experiment were larger, the total training time increases even though the network is able to converge in fewer epochs. Table VII displays the overall accuracy on the test set at each SNR for the identification problem. As expected, the network performs better on higher SNR transmissions, with significant performance drop-off below 20 dB. Table VIII shows the average over all devices of the ROC AUC and average precision-recall AUC at different SNRs for the verification networks.

V. CONCLUSION

In this paper we have demonstrated that CNN techniques which provide state-of-the-art performance in image and speech recognition problems can be applied to the problem of RF fingerprinting. Our method relies on steady state analysis of the signal and achieves high identification and verification accuracy on 7 ZigBee devices, and also permits use of the full transmission for fingerprinting regardless of the data content of the signal.

Deep learning methods are suitable for RF fingerprinting of the types of devices that may be used as CRN nodes because of their ability to adapt to the features available in the target signals. In contrast, approaches based on models or features that are assumed *a priori* may have difficulty detecting sufficient differences to distinguish the devices.

It is significant to note that the network used for lab data with multiple SNRs requires more parameters than the network used for outdoor data at approximately a fixed SNR. This demonstrates that increasing the size of the network and providing additional training data allows the network to learn additional channel effects while maintaining high classification performance. This shows promise for the ability of our technique to adapt to data collected under varying conditions, such as power levels and center frequencies by training on a more diverse dataset and using a larger network.

Because of the ability of our method to successfully classify devices at a single arbitrarily-chosen frequency, we believe that our method would perform equally well at any one center frequency. Because of this, we believe that our method could be applied directly to a cognitive radio network by training a larger network on data collected at a wide range of center frequencies for each device. A larger network with more parameters will have greater learning capacity to learn the patterns present at each possible center frequency, but will require a greater amount of data to train on.

In future work, we plan to extend our method to data captured from a larger set of devices. In addition, we plan to

evaluate our technique on transmissions with multiple power levels and channels. Finally, we will evaluate different classes of deep learning algorithms, such as recurrent neural networks and variational auto-encoders.

REFERENCES

[1] M. Youssef, M. Ibrahim, M. Abdelatif, L. Chen, and A. V. Vasilakos, "Routing metrics of cognitive radio networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 92–109, 2014.

[2] D. Cabric, S. M. Mishra, and R. W. Brodersen, "Implementation issues in spectrum sensing for cognitive radio," in *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004*, Pacific Grove, CA, Nov. 2004, pp. 772–776.

[3] R. Chen, J.-M. Park, and J. H. Reed, "Defense against primary user emulation attacks in cognitive radio networks," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 1, pp. 25–37, Jan. 2008.

[4] S. U. Rehman, K. W. Sowerby, and C. Coghill, "Radio-frequency fingerprinting for mitigating primary user emulation attack in low-end cognitive radios," *IET Communications*, vol. 8, no. 8, pp. 1274–1284, Jun. 2012.

[5] R. O. Afolabi, K. Kim, and A. Ahmad, "On secure spectrum sensing in cognitive radio networks using emitters electromagnetic signature," in *Proceedings of the 2009 Proceedings of 18th International Conference on Computer Communications and Networks*, San Francisco, CA, Sep. 2009, pp. 1–5.

[6] K. Kim, C. M. Spooner, I. Akbar, and J. H. Reed, "Specific emitter identification for cognitive radio with application to IEEE 802.11," in *IEEE Global Telecommunications Conference, 2008*, New Orleans, LO, Dec. 2008, pp. 1–5.

[7] P. F. Marshall, "Cognitive radio as a mechanism to manage front-end linearity and dynamic range," *IEEE Commun. Mag.*, vol. 47, no. 3, Mar. 2009.

[8] (2017) The zigbee alliance website. [Online]. Available: http://www.zigbee.org/zigbee-for-developers/applicationstandards/

[9] *IEEE Standard for Local and metropolitan area networks-Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE Standards Association Std., Rev. 2011, Sep. 2011.

[10] *ZigBee Specification*, ZigBee Alliance, Inc. Std., Rev. 20, Sep. 2012.

[11] T. Goodspeed, "Extracting keys from second generation zigbee chips," in *Black Hat USA 2009*, Las Vegas, NV, Jul. 2009.

[12] J. Wright, "Killerbee: Practical zigbee exploitation framework," in *ToorCon 11*, San Diego, CA, Oct. 2009.

[13] C. K. Dubendorfer, B. W. Ramsey, and M. A. Temple, "An RF-DNA verification process for zigbee networks," in *IEEE Military Communications Conference*, Orlando, FL, Oct./Nov. 2012.

[14] H. Patel, "Non-parametric feature generation for RF-fingerprinting on zigbee devices," in *Computational Intelligence for Security and Defense Applications (CISA), 2015 IEEE Symposium on*, Verona, NY, May 2015.

[15] T. J. Bihl, K. W. Bauer, and M. A. Temple, "Feature selection for RF fingerprinting with multiple discriminant analysis and using zigbee device emissions," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 8, pp. 1862–1874, Aug. 2016.

[16] C. Dubendorfer, B. Ramsey, and M. Temple, "Zigbee device verification for securing industrial control and building automation systems," in *International Conference on Critical Infrastructure Protection VII*, Washington, DC, Mar. 2013, pp. 47–61.

[17] O. Ureten and N. Serinken, "Wireless security through RF fingerprinting," *Canadian Journal of Electrical and Computer Engineering*, vol. 32, no. 1, pp. 27–33, 2007.

[18] Y. Yuan, Z. Huang, H. Wu, and X. Wang, "Specific emitter identification based on hilbert-huang transform-based time-frequency-energy distribution features," *IET Communications*, vol. 8, no. 13, pp. 2404–2412, Sep. 2014.

[19] C. Bertoncini, K. Rudd, B. Nousain, and M. Hinders, "Wavelet fingerprinting of radio-frequency identification (RFID) tags," *IEEE Trans. Ind. Electron.*, vol. 59, no. 12, pp. 4843–4850, Dec. 2012.

[20] N. Benvenuto, F. Piazza, and A. Uncini, "A neural network approach to data predistortion with memory in digital radio sistems," in *IEEE International conference on communication, 1993*, Geneva, Switzerland, May 1993, pp. 232–236.

[21] F. Mkadem and S. Boumaiza, "Physically inspired neural network model for RF power amplifier behavioral modeling and digital predistortion," *IEEE Trans. Microw. Theory Tech.*, vol. 59, pp. 913–923, Jan. 2011.

[22] H. Weng, X. Dong, X. Hu, D. G. Beetner, T. Hubing, and D. Wunsch, "Neural network detection and identification of electronic devices based on their unintended emissions," in *2005 International Symposium on Electromagnetic Compatibility*, Chicago, IL, Aug. 2005.

[23] G. B. Willson, "Radar classification using a neural network," in *Proceedings of SPIE 1294, Applications of Artificial Neural Networks*, Aug. 1990, pp. 200–210.

[24] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *International Conference on Engineering Applications of Neural Networks*, Aug. 2016.

[25] M. Schmidt, D. Block, and U. Meier, "Wireless interference identification with convolutional neural networks," *arXiv:1703.00737*, Mar. 2017.

[26] Y. Yuan, Z. Huang, and F. Wang, "Radio specific emitter identification based on nonlinear characteristics of signal," in *IEEE International Black Sea Conference on Communications and Networking, 2015*, Constanta, Romania, May 2015, pp. 77–81.

[27] C. Zhao, X. Wu, L. Huang, Y. Yao, and Y.-C. Chang, "Compressed sensing based fingerprint identification for wireless transmitters," *The Scientific World Journal*, vol. 2014, pp. 1–9, 2014.

[28] Digi XCTU. [Online]. Available: https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu

[29] A. C. Polak, S. Dolatshahi, and D. L. Goeckel, "Identifying wireless users via transmitter imperfections," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 7, pp. 1469–1479, Aug. 2011.

[30] A. C. Polak and D. L. Goeckel, "Wireless device identification based on RF oscillator imperfections," in *IEEE International Conference on Acoustic, Speech and Signal Processing*, Florence, Italy, May 2014, pp. 2679–2683.

[31] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," *CoRR*, vol. abs/1511.07289, 2015. [Online]. Available: http://arxiv.org/abs/1511.07289

[32] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-10)*, vol. 9, 2010, pp. 249–256.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[34] F. Chollet *et al.* Keras 2.0.9. [Online]. Available: https://github.com/fchollet/keras

[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. [Online]. Available: http://tensorflow.org/