Deep Learning Convolutional Neural Networks for Radio Identification

Shamnaz Riyaz, Kunal Sankhe, Stratis Ioannidis, and Kaushik Chowdhury
Electrical and Computer Engineering Department, Northeastern University, Boston, MA, USA
Email: mohammedriyaz.s@husky.neu.edu, sankhe.ku@husky.neu.edu, ioannidis@ece.neu.edu, krc@ece.neu.edu

Abstract—**Advances in software defined radio (SDR) technology allow unprecedented control on the entire processing chain, allowing modification of each functional block as well as sampling the changes in the input waveform. This paper describes a method for uniquely identifying a specific radio among nominally similar devices using a combination of SDR sensing capability and machine learning (ML) techniques. The key benefit of this approach is that ML operates on raw I/Q samples and distinguishes devices using only the transmitter hardware-induced signal modifications that serve as a unique signature for a particular device. No higher level decoding, feature engineering, or protocol knowledge is needed, further mitigating challenges of ID spoofing and coexistence of multiple protocols in a shared spectrum. The contributions of the paper are as follows: (i) The operational blocks in a typical wireless communications processing chain are modified in a simulation study to demonstrate RF impairments, which we exploit. (ii) Using an over-the-air dataset compiled from an experimental testbed of SDRs, an optimized deep convolutional neural network (CNN) architecture is proposed, and results are quantitatively compared with alternate techniques such as support vector machines and logistic regression. (iii) Research challenges for increasing the robustness of the approach, as well as the parallel processing needs for efficient training, are described. Our work demonstrates up to 90-99% experimental accuracy at transmitter-receiver distances varying between 2-50 feet over a noisy, multi-path wireless channel.**

I. Introduction

Emerging applications in the context of smart cities, autonomous vehicles, Internet of Things, and complex military missions, among others, require reconfigurability both at the systems and the protocol level within its communications architecture. These advances rely on a critical enabling component, namely, software defined radio (SDR): this allows cross-layer programmability of the transceiver hardware using high level directives. The promise of intelligent or so called *cognitive* radios builds on the SDR concept, where the radio is capable of gathering contextual information and adapting its own operation by changing the settings on the SDR based on what it perceives in its surroundings.

In many mission critical scenarios, problems in authenticating devices, ID spoofing and unauthorized transmissions are major concerns. Moreover, high bandwidth applications are causing a spectrum crunch, leading network providers to explore innovative spectrum sharing regimes in the TV whitespace and the sub-6GHz bands. In all of the above, identifying (i) the type of the protocol in use, and (ii) the specific radio transmitter (among many other nominally similar radios) become important. Our work on SDR-enabled device fingerprinting tackles these two scenarios by learning characteristic features of the transmitters in a pre-deployment training phase, which is then exploited during actual network operation. We recognize that SDRs come in diverse form factors with varying on-board computational resources. Thus, for general purpose use, any device fingerprinting approach must be computationally simple once deployed in the field. For this reason, we propose machine learning (ML) techniques, specifically, Deep Convolutional Neural Networks (CNNs), and experimentally demonstrate near-perfect radio identification performance in many practical scenarios.

• **Overview of our approach:** ML techniques have been remarkably successful in image and speech recognition, however, their utility for device level fingerprinting by feature learning has yet to be conclusively demonstrated. True autonomous behavior of SDRs, not only in terms of detecting spectrum usage, but also in terms of self-tuning a multitude of parameters and reacting to environmental stimulus is now a distinct possibility. We collect over $20 \cdot 10^6$ RF I/Q samples over multiple transmission rounds for each transmitter-receiver pair composed of off-the-shelf USRP SDRs. The SDRs transmit standards compliant IEEE 802.11ac physical layer waveforms, to create a database of received signals. These I/Q samples carry embedded signatures characteristic of different active transmitter hardware, but are also subject to alterations introduced by the wireless channel. The approach of providing raw time series radio signal by treating the complex data as dimension of 2 real valued I/Q inputs to the CNN, is motivated from modulation classification [1]. It has been found to be a promising technique for feature learning on large time series data. We develop a CNN architecture composed of multiple convolutional and max-pooling layers optimized for the task of radio fingerprinting. We partition the collected samples into separate instances and perform offline training on a computational cloud cluster, assigning weights to the inter-neuron connections. A holdout data set composed of totally unseen samples is used for estimation of detection accuracy.

• **Contributions and paper structure:** Our work makes the following key contributions. We survey and classify existing approaches in Sec. II. We design a simulation model of a typical wireless communications processing chain in MATLAB, and then modify the ideal operational blocks to demonstrate the RF impairments that we wish to learn in Sec. III. We describe the data gathering process for training the classifier in Sec. IV. We architect and experimentally validate an optimized deep convolutional neural network (CNN) for

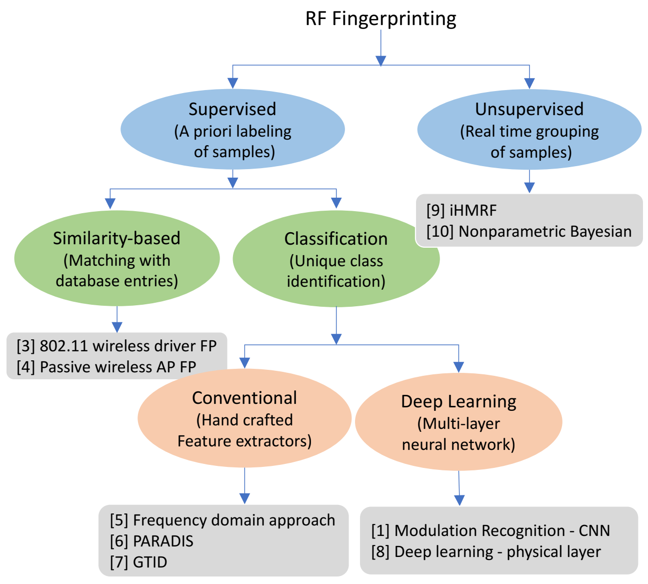

RF Fingerprinting
Supervised
(A priori labeling of samples)
Unsupervised
(Real time grouping of samples)
[9] iHMRF
[10] Nonparametric Bayesian
Similarity-based
(Matching with database entries)
Classification
(Unique class identification)
[3] 802.11 wireless driver FP
[4] Passive wireless AP FP
Conventional
(Hand crafted Feature extractors)
Deep Learning
(Multi-layer neural network)
[5] Frequency domain approach
[6] PARADIS
[7] GTID
[1] Modulation Recognition - CNN
[8] Deep learning - physical layer

Figure 1: RF Fingerprinting classification

radio fingerprinting in Sec. V, and quantitatively compare this approach with support vector machines and logistical regression in Sec. VI. Finally, research challenges for increasing the robustness of our approach are listed in Sec. VII and the conclusions are drawn in Sec. VIII. In summary, our CNN design demonstrates up to 90-99% experimental accuracy at transmitter-receiver distances varying between 2-50 feet over a noisy, multi-path wireless channel.

II. Related work

The key idea behind radio fingerprinting is to extract unique patterns (or features) and use them as signatures to identify devices. A variety of features at the physical (PHY) layer, medium access control (MAC) layer, and upper layers have been utilized for radio fingerprinting [2]. Simple unique identifiers such as IP addresses, MAC addresses, international mobile station equipment identity (IMEI) numbers can easily be spoofed. Location-based features such as radio signal strength (RSS) and channel state information (CSI) are susceptible to mobility and environmental changes. We are interested in studying those features that are inherent to a device's hardware, which are also unchanging and not easily replicated by malicious agents. We classify existing approaches in Fig. 1.

A. Supervised learning

This type of learning requires a large collection of labeled samples prior to network deployment for training the ML algorithm.

1) Similarity-based: Similarity measurements involve comparing the observed signature of the given device with the references present in a master database. In [3], a passive fingerprinting technique is proposed that identifies the wireless device driver running on an IEEE 802.11 compliant node by collecting traces of probe request frames from the devices. A supervised Bayesian approach is used to analyze the collected traces and generate the device driver fingerprint. [4] describes a passive blackbox-based technique, that uses TCP or UDP packet inter-arrival time to determine the type of access points using wavelet analysis. However these techniques rely on prior knowledge of vendor specific features.

2) Classification-based: There are several studies on supervised learning that exploit RF features such as I/Q imbalance, phase imbalance, frequency error, and received signal strength, to name a few. • *Conventional:* This form of classification examines a match with pre-selected features using domain knowledge of the system, i.e., the dominant feature(s) must be known a priori. [5] proposes classification by extracting the known preamble within a packet and computing spectral components. A set of log-spectral-energy features are given as input to the k-nearest neighbors (k-NN) discriminatory classifier. PARADIS [6] fingerprints 802.11 devices based on modulation-specific errors in the frame using SVM and k-NN algorithms with an accuracy of 99%. In [7], a technique for physical device and device-type classification called GTID using artificial neural networks is proposed. This method exploits variations in clock skews as well as hardware compositions of the devices. In general, as multiple different features are used, selecting the right set of features is a major challenge. This also causes scalability problems when large number of devices are present, leading to increased computational complexity in training. • *Deep Learning:* Deep learning offers a powerful framework for supervised learning approach. It can learn functions of increasing complexity, leverages large datasets, and greatly increases the the number of layers, in addition to neurons within a layer. [1] and [8] apply deep learning at the physical layer, specifically focusing on modulation recognition using convolutional neural networks. They classify 11 different modulation schemes. However, this approach does not identify a device, as we do here, but only the modulation type used by the transmitter.

B. Unsupervised learning

Unsupervised learning is effective when there is no prior label information about devices. In [9], an infinite Hidden Markov Random field (iHMRF)-based online classification algorithm is proposed for wireless fingerprinting using unsupervised clustering techniques and batch updates. Transmitter characteristics are used in [10] where a non-parametric Bayesian approach (namely, an infinite Gaussian Mixture Model) classifies multiple devices in an unsupervised, passive manner.

Transmitter identification using deep learning architectures is still in a nascent stage. Our work focuses on generation and processing of large number of RF I/Q samples to train the classifiers and eventually identify the devices uniquely.

III. Causes of hardware impairments

Using the MATLAB Communications System Toolbox, we simulate a typical wireless communications processing chain (see Fig. 2, with the shifts in the received complex valued I/Q samples), and then modify the ideal operational blocks to introduce RF impairments, typically seen in actual hardware

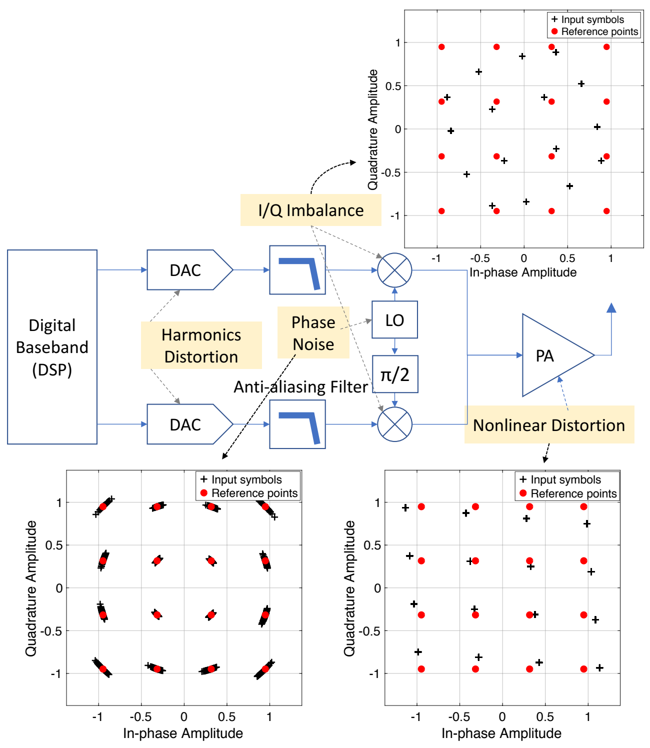


Figure 2: Typical transceiver chain with various sources of RF impairments.

implementations. This allows us to individually study the I/Q imbalance, phase noise, carrier frequency and phase offset, and nonlinearity of power amplifier, harmonic and power amplifier distortions.

•**I/Q imbalance:** Quadrature mixers that convert baseband to RF and vice versa are often impaired by gain and phase mismatches between the parallel sections of the RF chain dealing with the in-phase (I) and quadrature (Q) signal paths. The analog gain is never the same for each signal path and the difference between their amplitude causes amplitude imbalance. In addition, the delay is never exactly 90°, which causes phase imbalance.

•**Phase Noise**: The up-conversion of a baseband signal to a carrier frequency f_c is performed at the transmitter by mixing the baseband signal with the carrier signal. Instead of generating a pure tone at frequency f_c, i.e., $e^{j2\pi f_c t}$, the generated tone is actually $e^{j2\pi f_c t+\phi(t)}$, where $\phi(t)$ is a random phase noise. The phase noise introduces a rotational jitter. Phase noise is expressed in units of dBc/Hz, which represents the noise power relative to the carrier contained in a 1 Hz bandwidth centered at a certain offset from the carrier. Typical value of phase noise level is in the range $[-100,\ -48]$ dBc/Hz, with frequency offset in the range $[20,\ 200]$ Hz.

• **Carrier Frequency and Phase offset**: The performance of crystal oscillators used for generating the carrier frequency is specified with an accuracy in parts per million (ppm). The difference in transmitter and receiver carrier frequencies is referred to as carrier frequency offset.

•**Harmonic distortions**: The harmonics in a transmitted signal are caused by nonlinearities in the transmitter-side digital-to-analog converters. Harmonic distortion is measured in terms of total harmonic distortion, which is a ratio of the sum of the powers of all harmonic components to the power of the fundamental frequency of the signal. This distortion is usually expressed in either percent or in dB relative to the fundamental component of the signal.

•**Power amplifier distortions**: Power amplifier (PA) non-linearities mainly appear when the amplifier is operated in its non-linear region, i.e., close to its maximum output power, where significant compression of the output signal occurs. The distortions of the PA are generally modeled using AM/AM (amplitude to amplitude) and AM/PM (amplitude to phase) curves. The AM/AM causes amplitude distortion whereas AM/PM introduces phase shift. The nonlinearity of amplifier is modeled using Cubic Polynomial and Hyperbolic Tangent methods using Third-order input intercept point (IIP3) parameter. IIP3 expressed in (dBm) represents a scalar specifying the third order intercept.

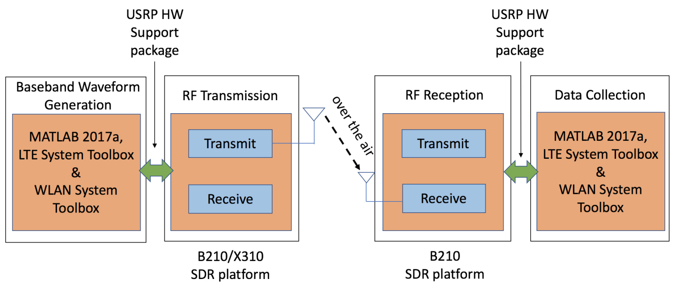


Figure 3: Data collection using SDR

IV. Data Collection for Deep Learning

A. *Experimental setup for Trace Data collection*

We study the performance of different learning algorithms, including linear support vector machine (SVM), logistic regression, and CNNs, using I/Q samples collected from an experimental setup of USRP SDRs, shown in Fig. 3. For the purpose of data collection at the receiver end, we use a fixed USRP B210. For the transmitter we use 5 different devices of the same family, i.e., USRP B210.

B. *Protocols of Operation*

We transmit different physical layer frames defined by the IEEE 802.11ac on each transmitter SDR. These frames are generated using MATLAB WLAN Systems toolbox, and are standards compliant. The data frames generated are random since we intend to transmit any data streams. These protocol frames are then streamed to the selected SDR for over-the-air wireless transmission. The receiving SDR samples the incoming signals at 1.92 MS/s sampling rate at center frequency of 2.45 GHz for WiFi. The collected complex I/Q samples are partitioned into subsequences. For our experimental study, we set a fixed subsequence length of 128, additional details of which are described in Sec. V-A2. Overall, we collect approximately 20 million samples for each of five SDRs.

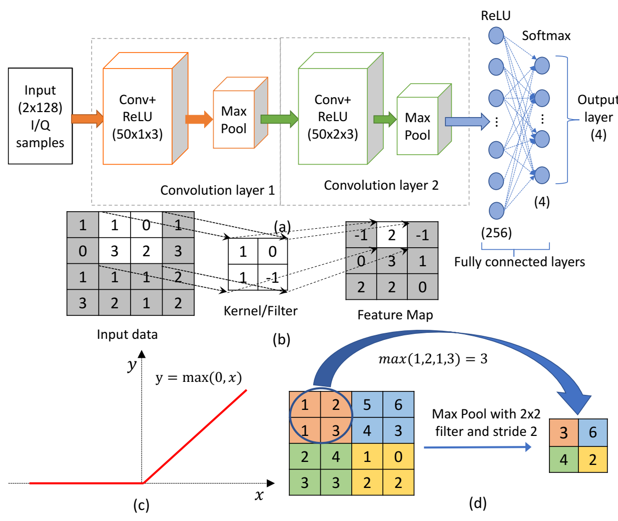


Figure 4: CNN architecture

C. Storage and Processing

The samples are further analyzed offline over (i) workstations with typical configurations of Core-i7 processor, 8GB RAM, and flash-based 512GB storage as well as (ii) Northeastern's Discovery cluster that has 16 compute nodes with a NVIDIA Tesla K40m GPU each. These nodes have 48 logical cores each, and on each node the GPU has 2880 CUDA computing cores. Each node has 128GByte of RAM configuration and dual Intel E5 2650 CPUs @ 2.00 GHz processor. These GPU servers are on a 10Gb/s TCP/IP backplane.

V. CNN based Radio Fingerprinting

The success of CNNs in vision and speech domains motivates our investigation in using CNNs for radio fingerprinting. The proposed method consists of two stages, i.e., a training stage and an identification stage. In the former, the CNN is trained using raw IQ samples collected from each SDR transmitter to solve a multi-class classification problem. In the identification stage, raw IQ samples of the unknown transmitter are fed to the trained neural network and the transmitter is identified based on observed value at the output layer. In this section, we first describe the CNN architecture and then present preprocessing of input data necessary to improve the performance.

A. CNN Architecture

Our CNN architecture is inspired in part by AlexNet [11], which shows remarkable performance in image recognition. As shown in the Fig. 4a, our network has four layers, which consists of two convolutional layers and two fully connected or dense layers. The input to the CNN is a windowed sequence of raw IQ samples with length 128. Each complex value is represented as two-dimensional real values, which results in the dimension of our input data growing to 2×128. This is then fed to the first convolution layer.

The convolution layer is the core building block of the CNN, whose primary purpose is to extract features from the input data. It consists of a set of spatial filters (also called *kernels*, or simply *filters*) that perform a convolution operation over input data. The operation of the convolution filter is shown with an example in Fig. 4b for intuitive understanding. A filter of size 2×2 is convolved with input data of size 4×4 by sliding across its dimension to produce two-dimensional *feature map*. A *stride* is the sliding interval of the filter and determines the dimension of the feature map. Our example shows stride 1 to produce a feature map of dimension 3×3. Each convolution layer consists of a set of such filters, which in turn operates independently to produce a set of two-dimensional feature maps. Our CNN architecture is composed of the convolution layer followed by an activation step that performs a pre-determined non-linear transformation on each element of the feature map. There are many possible activation functions, such as sigmoid and tanh; we use the Rectified Linear Unit (ReLU), as CNNs with ReLU train faster compared to alternatives. As shown in Fig. 4c, ReLU outputs $\max(x, 0)$ for an input x, replacing all negative values in the feature map by zero.

The convolution layer is generally followed by a pooling layer. Its functionality is to (a) introduce shift invariance (see also Sec. V-A3), as well as (b) reduce the dimensionality of the rectified feature maps of the preceding convolution layer, while retaining the most important information. We choose a pooling layer with filters of size 2×2 and stride 2, which downsamples the feature maps by 2 along both the dimensions. Among different filter operations (such as average, sum), max pooling gives better performance. As shown in Fig. 4d, max pooling of size 2×2 with stride 2 selects the maximum element in the non-overlapping regions (shown with different colors). Thus, it reduces the dimensionality of the feature map, which in turn reduces the number parameters and computations in the network.

The output of the second pooling layer is provided as input to the fully connected layer. A fully connected or dense layer is a traditional Multi Layer Perceptron (MLP), where the neurons have full connections to all activation steps in the previous layer, similar to regular neural networks. Its primary purpose is to perform the classification task on high-level features extracted from the preceding convolution layers. At the output layer, a *softmax* activation function is used. The classifer with softmax activation function gives probabilities (e.g. $[0.9, 0.09, 0.01]$ for three class labels).

Next, we discuss the selection hyperparameters of CNN to optimize the performance, followed by preprocessing of input data necessary for proper operation of CNN and finally shift-invariance property of our classifier.

1) Model Selection: We start with a baseline architecture consisting of two convolution layers and two dense layers, then progressively vary the hyperparameters to analyze their effect on the performance. The first parameter is the number of filters in the convolution layers. We observed that the number of filters within a range of $(30 - 256)$ provide reasonably similar performance. However, since the number of computations increases with an increase in the number of filters, we set 50 filters in both convolution layers for balancing the performance and computational cost. Similarly, we set 1×3 and 2×3 as the filter size in the first and second convolution layer respectively, since larger filter size does

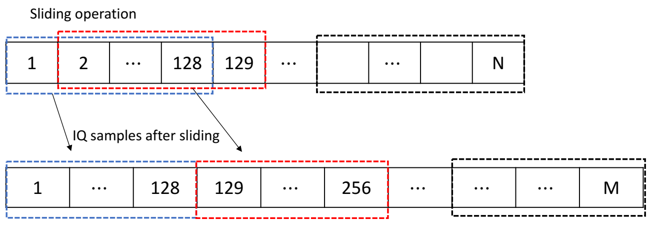


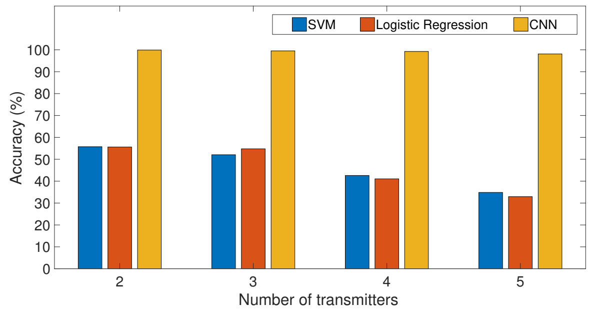
Figure 5: Sliding procedure

not offer significant performance improvement. Furthermore, increasing the number of convolution layers from 2 to 4 shows no improvement in the performance, which justifies continuation with two convolution layers. We then try to analyze the effect of the number of neurons in the first dense layer by varying it between 64 to 1024. Interestingly, we find that increasing the number of neurons beyond 256 does not improve the performance. Therefore, we set 256 neurons in the first dense layer. After finalizing the architecture and parameters of CNN, we carefully select the regularization parameters as follows: We use a dropout rate of 25% after first and second convolution layers and dropout of 50% at first dense layer. In addition, we use an ℓ_2 regularization parameter $\lambda = 0.0001$ to avoid over-fitting.

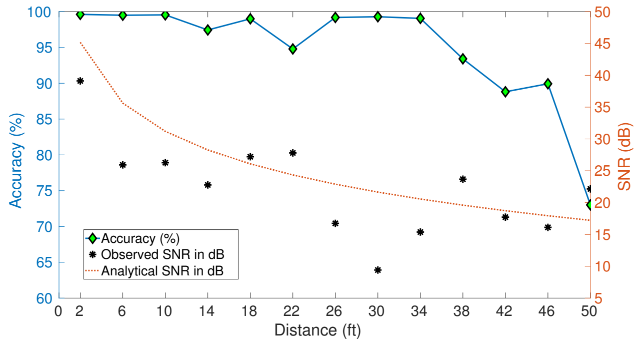
2) Preprocessing Data: Our experimental studies conducted on different representative classes of ML algorithms demonstrate significant performance improvement by choosing deep CNN. However, to ensure scalable performance over large number of devices, our CNN architecture needs to be modified. In addition, our input I/Q sequences, which represent a time-trace of collected samples, need to be suitably partitioned and augmented beyond a stream of raw I/Q samples.

Our classifiers operate on sequences of I/Q samples of a fixed length. In general, given sequences of length L, we can create $N = L/\ell$ subsequences of length ℓ by partitioning the input stream. We thus create $L - \ell$ subsequences by sliding a window of length ℓ over the larger sequence (or stream) of I/Q samples. Training classifiers over small subsequences leads to more training data points, which in turn yields a low variance but potentially high bias in the classification result. Conversely, large sequences may lead to high variance and low bias. We set 128 as sequence length. From a wireless communications viewpoint, the channel remains invariant in smaller durations of time. Hence, the ability to operate on smaller subsequences carved out of in-order received samples allows us to estimate the complex coefficients representing the wireless channel. Thus we train our classifiers over the input I/Q sequences by treating each real and imaginary part of a sample as two inputs, leading to a training vector of $2 \times \ell$ samples for a sequence of length ℓ.

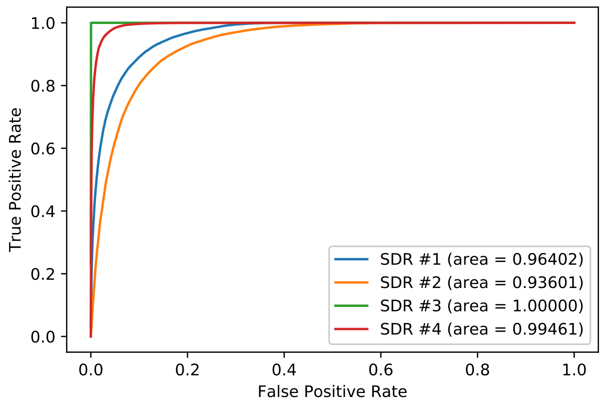
3) Shift Invariance: Another prominent characteristic of our CNN classifier both with respect to our final goal of identifying the transmitting device, but also in terms of feature extraction, is *shift invariance*. In short, all events described in Section III can occur at an arbitrary position in a given I/Q sequence. A classifier should be able to detect a device-specific impairment *irrespectively* of whether it occurs at e.g., the 1-st or 15-th position of an I/Q sequence. Convolved weights in each layer detect signals in arbitrary positions in the sequence, and a max-pool layer passes the presence of a signal to a higher layer irrespectively of where it occurs. To enhance the shift-invariance property of our classifier during training, we train it over sliding windows of length ℓ as shown in Fig. 5, rather than partitioned windows: this further biases the trained classifiers to shift-invariant configurations.



(a)



(b)



(c)

Figure 6: a) The accuracy comparison of SVM, logistic regression and CNN for $2-5$ devices using 5-fold cross-validation b) The plot of accuracy obtained using CNN for 4 devices over different distances between transmitter and receiver c) ROC curves for 4 devices under CNN classification

VI. Results and Performance Evaluation

We implement our CNN training and classifier in Keras running on top of TensorFlow on an NVIDIA Cuda enabled

Tesla K40m GPU. We evaluate the performance of our CNN classifier using 5-fold Cross Validation technique. We use StratifiedKFold class from the scikit-learn Python machine learning library to split up the training dataset into 5 folds. Our training set consists of $\approx 720K$ training examples and $\approx 80K$ examples for validation. We use another $200K$ examples for testing the performance of our trained model. Thus, we are able to obtain less biased estimate of the performance of our model. It took $\approx 43min$ to train our model. Performance evaluation on hold out dataset of $200K$ examples took only $\approx 3min$. The classifier output performance is measured using metrics such as accuracy and Area Under the Curve (AUC), the latter evaluated on the Receiver Operating Characteristic (ROC) curve comprising true positive rate on the Y-axis and false positive rate on the X-axis.

1) CNN vs. conventional algorithms: We first measure the performance of our dataset using SVM and logistic regression for the classification of nominally similar devices. We extract several features such as amplitude, phase and FFT values from the raw I/Q samples and built a rich set of features to train the classifiers. We obtain the classification accuracy for identification among 2, 3, 4 and 5 devices. As seen in Fig. 6a, accuracy measure with SVM and logistic regression algorithms for 2 devices is $\approx 55\%$ and it decreases further as the number of devices increases. The performance deterioration can be clearly seen in the Fig. 6a. We then train our CNN classifier using raw data to classify the same set of devices. With our deep CNN network, we are able to achieve accuracy 98% for five devices, as opposed to less than $\approx 33\%$ for the shallow learning SVM and logistic regression algorithms.

2) Impact of distance on radio fingerprinting: We run experiments to collect data over a distance ranging between 2-50 ft over steps of 4 ft, to evaluate the impact of distance (and possible multipath effect owing to reflections) on classification accuracy. Fig. 6b demonstrates the accuracy measure for the classification of 4 devices using CNN. It achieves classification accuracy greater than 95% up to the distance of 34ft. In addition, the observed SNR and analytical SNR (calculated using free-space path model) are shown in the same plot to elucidate the effect of received SNR on the classification accuracy. It is evident that the classification is robust against the fluctuations in SNR occurred due to path loss and multipath fading up to the distance of 34ft.

3) Receiver Operating Characteristics for radio fingerprinting: We obtained false positive rate and true positive rate to measure AUC. Fig. 6c shows the ROC curve for four similar WiFi devices. We can see that the CNN model works extremely well, as AUC ranges between 0.93 and 1. The AUC attained for each device is 0.964, 0.936, 1, and 0.994, respectively. This demonstrates CNN is the effective model for radio fingerprinting. Additionally, training our CNN network over a large dataset with Keras takes significantly lower time compared to any other aforementioned algorithms.

VII. Research Challenges

We now discuss the challenges associated with the implementation of CNNs for radio fingerprinting. In our experiments, we set the partition length as 128 through a rectangular windowing process. However, identifying the optimal length is a critical research objective and should be dependent on the channel coherence time. Varied CNN architectures may lead to significantly different results. Finding an optimal architecture which enhances device classification is an open research issue. A related challenge is obtaining the right balance between training time and the classification accuracy. Increasing the depth of the CNN beyond a point may not help the classification; in fact there are risks of overfitting the training set, as we found in some of our early experiments. Our work focuses on training the model with actual experimental data while a large body of earlier works attempt to solve a similar problem using synthetic data. There exists no standard dataset to benchmark the performance of our classifier, and releasing all datasets in widely accepted formats is essential for correct replication of experiments. Finally, as a future objective, our goal is to validate the performance of our classifier to identify large number of devices at distances of 100-200 ft. This may also require us to effect major changes in the architecture and find new optimum parameters.

VIII. Conclusion

We propose a radio fingerprinting approach based on deep learning CNN architecture to train using I/Q sequence examples. Our design enables learning features embedded in the signal transformations of wireless transmitters, and identifies specific devices. Furthermore, we have shown that our approach of device identification with CNN outperforms alternate ML techniques such as SVM, logistic regression for the identification of five nominally similar devices. Finally, we experimentally validate the performance of our design on a dataset collected over range of distances, 2 ft to 50 ft. We observe that detection accuracy decreases as the distance between transmitter and receiver increases and how computational resources such as Keras running with GPU support speed up the training time. Our future work involves increasing the robustness of the CNN architecture to allow scaling up to correct identification of 1000s of similar radios.

Acknowledgment

This work is supported by DARPA under the Young Faculty Award grant N66001-17-1-4042. We are grateful to Dr. Tom Rondeau, program manager at DARPA, for his insightful comments and suggestions that significantly improved the quality of the work.

References

[1] T. O'Shea, J. Corgan, and T. Charles Clancy, "Convolutional radio modulation recognition networks," 02 2016.

[2] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 94–104, Firstquarter 2016.

[3] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker, "Passive data link layer 802.11 wireless device driver fingerprinting," in *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, ser. USENIX-SS'06. Berkeley, CA, USA: USENIX Association, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267336.1267348

[4] K. Gao, C. Corbett, and R. Beyah, "A passive approach to wireless device fingerprinting," in *2010 IEEE/IFIP International Conference on Dependable Systems Networks (DSN)*, June 2010, pp. 383–392.

[5] I. O. Kennedy, P. Scanlon, F. J. Mullany, M. M. Buddhikot, K. E. Nolan, and T. W. Rondeau, "Radio transmitter fingerprinting: A steady state frequency domain approach," in *2008 IEEE 68th Vehicular Technology Conference*, Sept 2008, pp. 1–5.

[6] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, ser. MobiCom '08. New York, NY, USA: ACM, 2008, pp. 116–127. [Online]. Available: http://doi.acm.org/10.1145/1409944.1409959

[7] S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah, "Gtid: A technique for physical device and device type fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 519–532, Sept 2015.

[8] T. J. O'Shea and J. Hoydis, "An introduction to machine learning communications systems," *CoRR*, vol. abs/1702.00832, 2017. [Online]. Available: http://arxiv.org/abs/1702.00832

[9] F. Chen, Q. Yan, C. Shahriar, C. Lu, W. Lou, and T. C. Clancy, "On passive wireless device fingerprinting using infinite hidden markov random field," *submitted for publication*.

[10] N. T. Nguyen, G. Zheng, Z. Han, and R. Zheng, "Device fingerprinting to enhance wireless security using nonparametric bayesian method," in *2011 Proceedings IEEE INFOCOM*, April 2011, pp. 1404–1412.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. USA: Curran Associates Inc., 2012, pp. 1097–1105. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999134.2999257