DeepRadioID: Real-Time Channel-Resilient Optimization of Deep Learning-based Radio Fingerprinting Algorithms

Francesco Restuccia
Northeastern University
Boston, MA, USA
frestuc@northeastern.edu

Salvatore D'Oro
Northeastern University
Boston, MA, USA
s.doro@northeastern.edu

Amani Al-Shawabka
Northeastern University
Boston, MA, USA
amani@northeastern.edu

Mauro Belgiovine
Northeastern University
Boston, MA, USA
belgiovine@northeastern.edu

Luca Angioloni
Northeastern University
Boston, MA, USA
angioloni@northeastern.edu

Stratis Ioannidis
Northeastern University
Boston, MA, USA
ioannidis@northeastern.edu

Kaushik Chowdhury
Northeastern University
Boston, MA, USA
krc@northeastern.edu

Tommaso Melodia
Northeastern University
Boston, MA, USA
melodia@northeastern.edu

ABSTRACT

Radio fingerprinting provides a reliable and energy-efficient IoT authentication strategy by leveraging the unique hardware-level imperfections imposed on the received wireless signal by the transmitter's radio circuitry. Most of existing approaches utilize hand-tailored protocol-specific feature extraction techniques, which can identify devices operating under a pre-defined wireless protocol only. Conversely, by mapping inputs onto a very large feature space, deep learning algorithms can be trained to fingerprint large populations of devices operating under any wireless standard.

One of the most crucial challenges in radio fingerprinting is to counteract the action of the wireless channel, which decreases fingerprinting accuracy significantly by disrupting hardware impairments. On the other hand, due to their sheer size, deep learning algorithms are hardly re-trainable in real-time. Another aspect that is yet to be investigated is whether an adversary can successfully impersonate another device's fingerprint. To address these key issues, this paper proposes *DeepRadioID*, a system to optimize the accuracy of deep-learning-based radio fingerprinting algorithms *without retraining the underlying deep learning model*. The key intuition is that through the application of a carefully-optimized digital finite input response filter (FIR) at the transmitter's side, we can apply tiny modifications to the waveform to strengthen its fingerprint according to the current channel conditions. We mathematically formulate the *Waveform Optimization Problem* (WOP) as the problem of finding, for a given trained neural network, the optimum FIR to be used by the transmitter to improve its fingerprinting accuracy.

We extensively evaluate *DeepRadioID* on a experimental testbed of 20 nominally-identical software-defined radios, as well as on two datasets made up by 500 ADS-B devices and by 500 WiFi devices provided by the DARPA RFMLS program. Experimental results show that *DeepRadioID* (i) increases fingerprinting accuracy by about 35%, 50% and 58% on the three scenarios considered; (ii) decreases an adversary's accuracy by about 54% when trying to imitate other device's fingerprints by using their filters; (iii) achieves 27% improvement over the state of the art on a 100-device dataset.

CCS CONCEPTS

• **Computer systems organization** → **Embedded and cyber-physical systems**; • **Security and privacy** → **Mobile and wireless security**; • **Networks** → *Network experimentation.*

KEYWORDS

Radio Fingerprinting, Deep Learning, Security, Optimization, Testbed

ACM Reference Format:
Francesco Restuccia, Salvatore D'Oro, Amani Al-Shawabka, Mauro Belgiovine, Luca Angioloni, Stratis Ioannidis, Kaushik Chowdhury, and Tommaso Melodia. 2019. *DeepRadioID*: Real-Time Channel-Resilient Optimization of Deep Learning-based Radio Fingerprinting Algorithms. In *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'19).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Thanks to its unprecedented pervasiveness, one the most crucial issues in the Internet of Things (IoT) is designing *scalable, reliable and energy-efficient authentication mechanisms* [13, 22]. However, most of the existing authentication mechanisms are not well-suited to the IoT since they are heavily based on cryptography-based algorithms and protocols, which are often too computational expensive to be run on tiny, energy-constrained IoT devices [16].

To address this key issue, a number of techniques based on radio fingerprinting have been proposed over the last few years

MobiHoc'19, July 2019, Catania, Italy

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

[1, 11, 12, 18–20]. The core intuition behind radio fingerprinting is that wireless devices usually suffer from small-scale hardware-level imperfections typically found in off-the-shelf RF circuitry, such as phase noise, I/Q imbalance, frequency and sampling offset, and harmonic distortions [9]. We can thus obtain a "fingerprint" of a wireless device by estimating the RF impairments on the received waveform and associating them to a given device [21].

Traditional techniques for radio fingerprinting (which are discussed in details in Section 2) rely on complex feature-extraction techniques that leverage protocol-specific characteristics (such as WiFi pilots/training symbols [1, 18] or ZigBee O-QPSK modulation [12]) to extract hardware impairments. Therefore, they are not general-purpose in nature and are hardly applicable to the IoT, where a plethora of different wireless protocols are used [23]. To overcome this limitation, in this paper we use techniques based on *deep learning* [8] to design *general-purpose, high-performance, and optimizable* radio fingerprinting algorithms. Thanks to the very large number of parameters (*i.e.*, in the order of 10^6 or more), deep neural networks can analyze unprocessed I/Q samples without the need of application-specific and computational-expensive feature extraction and selection algorithms [14].

Challenges. There are a number of critical issues in applying deep learning techniques to RF fingerprinting. First, deep learning models usually require a significant time to be re-trained, even with modern GPUs [2]. Therefore, we cannot assume that *the underlying deep learning model can be retrained in real time*. Second, a fingerprinting system must evaluate *the impact of adversarial actions*. Specifically, to the best of our knowledge, existing work has not yet evaluated if and when an adversary can imitate a legitimate device's fingerprint. Last, but not least, we need to *address the (potentially disruptive) action of the wireless channel on the system's fingerprinting accuracy*. This is because, due to channel action, two identical waveforms transmitted by the same RF interface at two different moments in time are usually different from each other. This implies that the models will operate on *non-stationary* input data [6], which significantly decreases the model's fingerprinting accuracy when the classifier is used on data collected with a wireless channel that is significantly different from the one used to train it.

To illustrate this crucial point, Figure 1 shows the confusion matrices of a deep learning model trained to fingerprint 5 devices through the experimental testbed that will be presented in Section 6. The confusion matrix (a) was computed on data collected approximately 5 minutes after the training data was collected, while Figure 1(b) was obtained by testing the model on completely new data collected 7 days after the model was trained. Figure 1 remarks that the fingerprinting accuracy decreases significantly when data collected under completely different channel conditions is fed to the model, demonstrating that the channel's action must indeed be addressed through real-time optimization.

Novel Contributions. This paper addresses the above challenges by making the following novel contributions:

• We propose *DeepRadioID*, a system for real-time channel- and adversary-resilient optimization of deep-learning-based radio fingerprinting algorithms. The key innovation behind *DeepRadioID* is to leverage a carefully-optimized digital finite input response filter (FIR) at the transmitter's side, which slightly modifies its baseband signal to compensate for current channel conditions. The optimal FIR is computed by the receiver and sent back as feedback to the transmitter. We postulate the *Waveform Optimization Problem* (WOP) to find the optimal FIR, and derive a novel algorithm based on the *Nonlinear Conjugate Gradient* (NCG) method to efficiently solve it. We show in Section 5 that the FIR's action can be effectively compensated at the receiver's side through the discrete Fourier transform (DFT) of the received signal, thus causing a negligible throughput decrease (*i.e.*, less than 0.2% in our experiments).

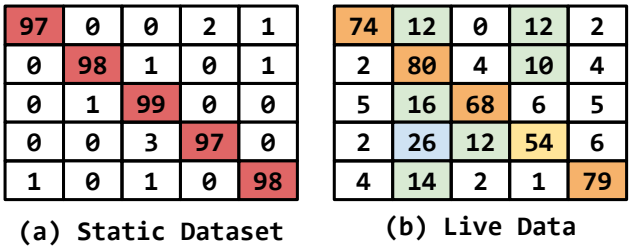

Figure 1: Confusion matrices of 5-device bit-similar model with (a) original dataset; (b) live-collected data. The figure highlights that different wireless channel conditions imply a loss in fingerprinting accuracy.

• We extensively evaluate the performance of *DeepRadioID* on an experimental testbed made up of 20 bit-similar devices (*i.e.*, transmitting the same baseband signal through nominally-identical RF interfaces and antennas). To evaluate the scalability of *DeepRadioID* and to experiment with different wireless technologies and deeper learning models, we also leverage two datasets of IEEE 802.11a/g (WiFi) and Automatic Dependent Surveillance – Broadcast (ADS-B) transmissions, each containing 500 devices. These transmissions were collected "in the wild" by DARPA for the RFMLS program. To the best of our knowledge, we are the first ever to evaluate radio fingerprinting algorithms on datasets of such dimension. Experimental results indicate that (i) an adversary trying to imitate a legitimate device's fingerprint by using the same FIR filter *decreases* its fingerprinting accuracy by about 54% on the average; (ii) DeepRadioID *increases* the fingerprinting accuracy by (a) 35% on our bit-similar experimental testbed, and (b) by 50% and by 58% on the 500-device ADS-B and WiFi datasets, respectively; (iii) by comparing with the state of the art [18], *DeepRadioID* improves the accuracy by about 27% on a reduced dataset of 100 WiFi devices.

2 RELATED WORK

Radio fingerprinting has received significant attention over the last few years – for an excellent survey paper on the topic, the reader may refer to [21].

The vast majority of existing work has applied carefully-tailored feature extraction techniques at the physical layer to fingerprint wireless devices [1, 11, 12, 18–20]. Nguyen *et al.* [11] use device-dependent channel-invariant radio-metrics and propose a non-parametric Bayesian method to detect the number of devices. However, the effectiveness of the features is proven with a testbed made up of only four ZigBee transmitters. Brik *et al.* [1] considered a combination of frequency offset, transients, and constellation errors

to fingerprint 130 IEEE 802.11b cards with an accuracy of 99%. However, conversely from ours, the experiments in [1] were performed in an RF-insulated environment (*i.e.*, without any channel effect), thus the algorithms' effectiveness in real-world environments has yet to be established. More recently, Vo *et al.* [18] proposed a series of algorithms with features based on frequency offsets, transients and the WiFi scrambling seed, and validated them with off-the-shelf WiFi cards in a non-controlled RF environment, achieving accuracy between 44 and 50% on 93 devices. In Section 6.4, we show that *DeepRadioID* improves the acccuracy of 27% on a 100-device WiFi testbed. Recently, Peng *et al.* [12] proposed fingerprinting algorithms for ZigBee devices based on modulation-specific features such as differential constellation trace figure (DCTF), showing that their features achieve almost 95% accuracy on a 54-radio testbed.

The key drawback of existing feature-based fingerprinting techniques is that they are inherently tailored for a specific wireless technology (*e.g.*, WiFi or ZigBee), which ultimately limits their applicability to IoT scenarios where devices operate under different standards [23]. Moreover, existing work on feature-based fingerprinting has not considered the problem of optimizing the algorithm's accuracy in real-time. Thus, we consider *deep learning* to design a general-purpose and scalable fingerprinting system. Although watermarking has been proposed to identify devices in the IoT [5], it requires the insertion of additional information, which *DeepRadioID* avoids.

The closest work to ours is [15], where the authors proposed the usage of convolutional neural networks to fingerprint nominally-identical USRP X310 devices. They also show that by using artificially introduced hardware impairments at the transmitter's side, the accuracy can be improved to 99%. However, [15] suffers from the following key limitations: (i) the artificial impairments cannot be accurately compensated at the receiver's side; (ii) the relationship between hardware impairment and accuracy is not fully characterized; and (iii) adversarial action is not considered. *DeepRadioID* overcomes the above limitations by proposing a system where we increase in real-time the fingerprinting accuracy through the application of a FIR filter that (a) is obtained through rigorous optimization (Section 4.3); (b) can be compensated at the receiver's side (Section 5); (c) cannot be used by an adversary to impersonate another device (Section 6).

3 DeepRadioID: AN OVERVIEW

We first discuss some key observations and motivations to motivate our design choices in Section 3.1. We then provide an in-depth description and a walk-through of the main steps involved in the fingerprinting process in Section 3.2.

3.1 DeepRadioID: Key Intuitions

The need to optimize the accuracy of fingerprinting systems arises from the fact that the wireless channel is dynamic and almost unpredictable in nature. Thus, hardware impairments such as I/Q imbalances, DC offset, phase noise, carrier/sampling offsets, and power amplifier distortions can be disrupted by the channel's action. Moreover, these impairments are also time-varying and dependent on a number of factors, such as local oscillator (LO) frequency [17] and current temperature of the RF circuitry [9]. These considerations imply that we cannot assume impairments as perfectly stationary – hence the need for real-time optimization.

To address the non-stationary nature of the problem, our first observation is that convolutional neural networks (CNNs) have shown to be prodigiously suited to recognize complex "patterns" in input data [10] – these patterns are, in our case, the imperfections in the radio hardware. However, a major challenge that still lingers is *how do we optimize the CNN's output for a given device without retraining the CNN itself.* To answer this question, we devise a new approach based on finite impulse response (FIR) filtering of the transmitter's baseband signal to "restore" the patterns that are disrupted by the current channel conditions – thus making the signal "more recognizable" to the CNN. We use FIRs because of the following: (i) FIRs are very easy to implement in both hardware and software on almost any wireless device; (ii) the computation complexity of applying a FIR filter of length m to a signal is $O(m)$ – thus it is a very efficient algorithm; and most importantly, (iii) its effect on the BER can be almost perfectly compensated at the receiver's side, as shown in Section 5.

However, this approach spurs another challenge, which is *how to set the FIR taps in such a way that the fingerprinting accuracy for a given device is maximized.* Our intuition here is to find the FIR that modifies input x so that the resulting x^* signal maximizes the neuron activation correspondent to a given device, as shown in Section 4.3. We are able to do this efficiently since the layers inside CNNs, although non-linear, are derivable, and thus we can compute the gradient of the output with respect to the FIR taps according to a given input. This way, we can design an optimization strategy that is fundamentally general-purpose in nature.

3.2 DeepRadioID: A Walk-Through

Figure 2 provides a walk-through of the main building blocks of *DeepRadioID* and the main operations involved in the fingerprinting process. We highlight with a shade of blue the blocks that are added to the normal modulation/demodulation chain as part of *DeepRadioID*. The walk-through also shows how an adversary may try to imitate another device's fingerprint. The detailed explanation of *DeepRadioID*'s main module will be given in Section 4.

The first step for a legitimate device "A" that wants to be authenticated by a receiver "R" is to filter its baseband signal with FIR $\boldsymbol{\phi}_{A,i-1}$ (step 1), which was obtained at the previous optimization step. FIR $\phi_{A,0}$ is set to 1 (*i.e.*, no filtering). The filtered signal is then sent to A's RF interface (step 2). By also accounting the effect of the wireless channel, "R" will receive a baseband signal $\mathbf{z}_{AR} = \mathbf{x}_A \circledast \boldsymbol{\phi}_{A,i-1} \circledast \mathbf{h_{AR}} + \mathbf{w_{AR}}$, where $\mathbf{x}_A$ is the transmitted symbol sequence, $\mathbf{h_{AR}}$ and $\mathbf{w}_{AR}$ are the fading and noise introduced by the channel, respectively. The I/Q samples of $\mathbf{z}_{AR}$ are then fed to a CNN to fingerprint the originating device (step 4). The fingerprinting result is then used to compensate the FIR filter $\boldsymbol{\phi}_{A,i-1}$ (step 5, discussed in Section 5), so that the resulting signal is then sent to the symbol demodulation logic to recover the application's data (step 6). The I/Q samples and the fingerprinting result are then fed to the *DeepRadioID* FIR Optimization module (step 7, presented in Section 4). The optimal FIR filter $\boldsymbol{\phi}_{A,i}$ is then sent back to A to improve its fingerprinting accuracy (step 8).

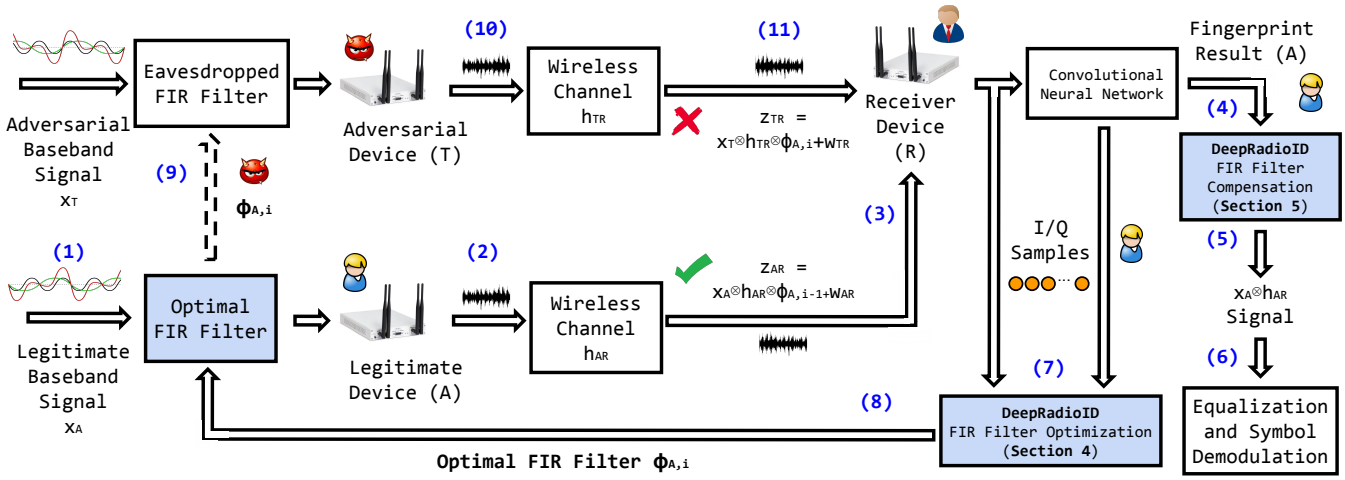

Figure 2: A high-level overview of the *DeepRadioID* system, where we also illustrate an adversary (T) trying to impersonate a legitimate device (A) using an eavesdropped FIR filter. Since A's FIR filter has been tailored to match A's unique channel and impairment conditions, we show in Section 6 that T does not improve its fingerprinting accuracy by using A's filter.

We now examine the case of adversarial action as follows. We assume that an adversarial device "T" is capable of eavesdropping A's FIR $\boldsymbol{\phi}_{A,i}$. T's target here is to impersonate A by spoofing A's hardware fingerprint (step 9). After T's baseband signal is transmitted and goes through the wireless channel (step 10), the baseband signal received by R will be $\mathbf{z_{TR}} = \mathbf{x_R} \circledast \boldsymbol{\phi}_{\mathbf{A,i-1}} \circledast \mathbf{h_{TR}} + \mathbf{w_{TR}}$, which is then fed to the CNN as input. However, we show in Section 6 that, since A's FIR filter has been optimized for A's unique hardware impairments and A's current wireless channel, T will not be able to imitate A's fingerprint by using A's FIR filter.

4 DeepRadioID FIR OPTIMIZATION

In this section, we describe in details the *DeepRadioID* FIR Filter Optimization module. We first provide some background notions in Section 4.1, followed by our FIR-based waveform modification approach in Section 4.2. We then introduce the Waveform Optimization Problem (WOP) in Section 4.3

4.1 Background Notions and Definitions

Let us define as *input* a set of N consecutive I/Q samples that constitute an input to the classifier. Let us also define as *slice* a set of S inputs, and as batch a set of B slices. Let us label the D devices being classified with a label between 1 and D. We model the classifier as as a function $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{C}^N$ and $\mathcal{Y} \subseteq \mathbb{R}^D$ represent respectively the spaces of the classifier's input (*i.e.*, an example) and output (*i.e.*, a probability distribution over the set of D devices). Specifically, the output of the classifier can be represented as a vector $(f_1, f_2, ..., f_D) \in \mathcal{Y}$, where the i-th component denote the probability that the input fed to the CNN belongs to device i.

DeepRadioID relies on discrete causal finite impulse response filters (in short, FIRs) to achieve real-time adaptive waveform modification. FIRs present several advantages – first, causal filters do not depend on future inputs, but only on past and present ones. Second, they can be represented as a weighted and finite term sum, which allows to *accurately predict the output of the FIR for any given input*. More formally, a FIR is described by a finite sequence $\boldsymbol{\phi}$ of M *filter taps*, i.e., $\boldsymbol{\phi} = (\phi_1, \phi_2, \ldots, \phi_M)$. For any input $\mathbf{x} \in \mathcal{X}$, the filtered n-th element $\hat{x}[n] \in \hat{\mathbf{x}}$ can be written as

$$\hat{x}[n] = \sum_{j=0}^{M-1} \phi_j x[n-j] \tag{1}$$

Since both wireless channel and hardware impairments operate in the complex domain by rotating and amplifying/attenuating the amplitude of the signal, *we can manipulate the position in the complex plane of the transmitted I/Q symbols.* By using complex-valued filter taps, *i.e.*, $\phi_k \in \mathbb{C}$ for all $k = 0, 1, \ldots, M-1$, we can rewrite Eq. (1) as follows:

$$\begin{aligned} \hat{x}[n] &= \sum_{k=0}^{M-1} (\phi_k^R + j\phi_k^I)(x^R[n-k] + jx^I[n-k]) \\ &= \hat{x}^R[n] + j\hat{x}^I[n] \end{aligned} \tag{2}$$

where $x_k^R[n] = \mathrm{Re}\{x_k[n]\}$, $x_k^I[n] = \mathrm{Im}\{x_k[n]\}$, $\phi_k^R = \mathrm{Re}\{\phi_k\}$ and $\phi_k^I = \mathrm{Im}\{\phi_k\}$. Eq. (2) clearly shows that it is possible to manipulate the input sequence by filtering each I/Q sample. For example, to rotate all I/Q samples by $\theta = \pi/4$ radiants and halve their amplitude, we may set $\phi_1 = \frac{1}{2}\exp^{j\frac{\pi}{4}}$ and $\phi_k = 0$ for all $k > 1$. Similarly, other complex manipulations can be obtained by fine-tuning filter taps.

4.2 FIR-based Waveform Modification

Although channel equalization can effectively reduce the effect of channel distortions on the position of the received I/Q samples, the algorithms involved are generally not perfect and only *partially* counteract phase and amplitude variations caused by the channel. For this reason, we must devise techniques to dynamically adapt to rapidly changing channel conditions (*e.g.*, fast-fading/multi-path) and thus improve the fingerprinting accuracy for a given device.

DeepRadioID leverages FIR filters to maximize the accuracy of the classifier by dynamically counterbalancing inaccurate channel

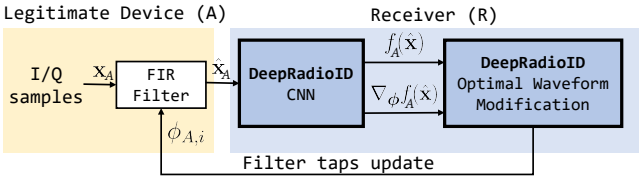

Figure 3: Waveform modification optimization loop.

equalization. Figure 3 shows a block diagram of the waveform modification optimization loop performed by *DeepRadioID*. Specifically, we add a FIR filter before the first CNN layer. This additional layer uses FIRs to manipulate the input example according to Eq. (2). The corresponding output sequence is then fed to the CNN.

As shown in Figure 2, let A be the target device for which we want to improve the detection accuracy of the CNN, and let $\boldsymbol{\phi}_{A,i}$ be the filter taps associated to the target device at the i-th optimization step. By using the filtering-based waveform modification on the input sequence $\mathbf{x}$, the output $f_A(\hat{\mathbf{x}}) \in \mathcal{Y}$ of the classifier with respect to the filtered sequence $\hat{\mathbf{x}}$ can be written as a function of the filter taps $\boldsymbol{\phi}_{A,i}$. Specifically, we have that

$$f_A(\hat{\mathbf{x}}) = f_A(\mathbf{x}, \boldsymbol{\phi}_{\mathbf{A},\mathbf{i}}) \tag{3}$$

Eq. (3) clearly shows that the accuracy of the classifier depends on the actual FIR tap vector $\boldsymbol{\phi}_{A,i}$. Thus, we are interested in devising mechanisms to optimally manipulate $\boldsymbol{\phi}_{A,i}$ such that (i) the classification accuracy for the target device is maximized (Section 4.3); and (ii) the waveform modification does not negatively impact the BER of data transmission activities (Section 5). To simplify the notation, henceforth we will remove the i subscript.

4.3 Waveform Optimization Problem (WOP)

We can now formally define the objective of *DeepRadioID* as follows: (i) maximize the accuracy of the classifier for a specific target device A; and (ii) to guarantee that the resulting BER does not exceed a given maximum tolerable threshold $\text{BER}_{\max}$. Since we aim at achieving channel-resilient waveform modification, we need to compute a FIR parameter configuration $\boldsymbol{\phi}_A$ that can be used for multiple consecutive transmissions. It is worth mentioning that to compute different $\boldsymbol{\phi}_A$ values for each single input $\mathbf{x}$ is inefficient in many cases. Indeed, the obtained FIR would be effective for the considered input only, *i.e.*, if applied to another input sequence $\mathbf{x}' \neq \mathbf{x}$, the FIR might decrease the accuracy of the classifier. Thus, maximizing the accuracy with respect to a single input $\mathbf{x}$ might result in poor performance.

To overcome the above problem, rather than maximizing the accuracy of the classifier on an input-by-input basis, we compute the the FIR $\boldsymbol{\phi}_A$ that maximizes the activation probability f_A over a set of S consecutive inputs, *i.e.*, a slice.

The Waveform Optimization Problem (WOP) can be then defined as follows:

$$\underset{\boldsymbol{\phi}}{\text{maximize}} \quad \frac{1}{S}\sum_{s=1}^{S} f_A(\mathbf{x}_s, \boldsymbol{\phi}) \tag{WOP}$$

$$\text{subject to } \text{BER}_A(\mathbf{x}_s, \boldsymbol{\phi}) \leq \text{BER}_{\max}, \quad \forall s = 1, 2, \ldots, S \tag{C1}$$

where the objective function represents the per-slice average activation probability for device A, $\mathbf{x}_s$ is the s-th input of the slice, and the $\text{BER}_A(\cdot)$ represents the BER function corresponding to transmissions from target node A.

It is worth mentioning that the function $f_A(\mathbf{x}_s, \boldsymbol{\phi})$ represents the CNN, and thus it outputs the probability that the input I/Q samples $\mathbf{x}$ belong to device A. Thus, by solving the WOP, we compute a FIR that maximizes the activation probability of the neuron associated to A.

Problem (WOP) is significantly challenging because (i) the function f_A is CNN-specific and depends from a very high number of parameters (generally in the order of millions), it is highly non-linear and to the best of our knowledge, there are no mathematical closed-form expressions for such a function, even for relatively small CNNs; (ii) the maximum BER constraint (C1) depends from numerous device-specific parameters (*e.g.*, modulation, coding, transmission power and SNR) and it is generally non-linear.

Notwithstanding the above challenges, and as we will discuss in detail in Section 5, the impact of the waveform modification procedure on the BER of communications among the receiver and the target device A is negligible. In fact, as shown in Figure 2, *DeepRadioID* embeds a FIR Filter Compensation module that uses peculiar features of FIR filters, *e.g.*, their Fourier transform, to successfully reconstruct the original transmitted unfiltered sequence of I/Q symbols. This compensation procedure effectively removes any coupling between waveform modification procedures and BER, *i.e.*, $\text{BER}_A(\mathbf{x}, \boldsymbol{\phi}) \approx \text{BER}_A(\mathbf{x})$. Accordingly, it is possible to relax Constraint (C1) by removing it from the optimization problem (WOP).

The relaxed WOP can be formulated as

$$\underset{\boldsymbol{\phi}}{\text{maximize}} \quad \sum_{s=1}^{S} f_A(\mathbf{x}_s, \boldsymbol{\phi}) \tag{RWOP}$$

where we have also omitted the constant term $1/S$.

4.3.1 Solving the RWOP. As already mentioned, f_A is non-linear and generally does not possess any useful property in terms of monotonicity, concavity and existence of a global maximizer. However, for any input $\mathbf{x}$ of the slice, by using *back-propagation* and the chain rule of derivatives it is possible to let the CNN compute the gradient $\nabla_{\hat{\mathbf{x}}} f_A(\hat{\mathbf{x}})$ of the classification function f_A with respect to the filtered input sequence $\hat{\mathbf{x}}$. It is worth noting that $\nabla_{\hat{\mathbf{x}}} f_A(\hat{\mathbf{x}})$ shows how different input sequences affect the accuracy of the classification function. Nevertheless, we are interested in evaluating the gradients $\nabla_{\boldsymbol{\phi}} f(\hat{\mathbf{x}})$ to predict how the accuracy of the classifier varies as a function of the FIR filtering function. Hence, we need to extend back-propagation to the waveform modification block.

From Eq. (2), $\hat{\mathbf{x}}$ is a function of $\boldsymbol{\phi}$, thus the gradient of f_A with respect to the filter taps $\boldsymbol{\phi}$ can be computed as

$$\nabla_{\boldsymbol{\phi}} f_A(\hat{\mathbf{x}}) = J_{f_A}(\boldsymbol{\phi})^{\top} \cdot \nabla_{\hat{\mathbf{x}}} f_A(\hat{\mathbf{x}}) \tag{4}$$

where $J_{f_A}(\boldsymbol{\phi})$ is the Jacobian matrix of $f_A(\mathbf{x}, \boldsymbol{\phi})$ with respect to $\boldsymbol{\phi}$, $\top$ is the transposition operator, and $\cdot$ stands for matrix dot product.

From Eq. (4) and Eq. (2), each element in $\nabla_{\boldsymbol{\phi}} f_A(\hat{\mathbf{x}})$ can be written as

$$\frac{\partial f_A(\mathbf{x}, \boldsymbol{\phi})}{\partial \phi_k^Z} = \sum_{n=1}^{N} \left(\frac{\partial f_A(\mathbf{x}, \boldsymbol{\phi})}{\partial \hat{x}^R[n]} \frac{\partial \hat{x}^R[n]}{\partial \phi_k^Z} + \frac{\partial f_A(\mathbf{x}, \boldsymbol{\phi})}{\partial \hat{x}^I[n]} \frac{\partial \hat{x}^I[n]}{\partial \phi_k^Z} \right) \tag{5}$$

where $k = 0, 1, \ldots, M-1$, N is the length of the input sequence and $Z \in \{R, I\}$.

By using Eq. (2), $\frac{\partial \hat{x}^R[n]}{\partial \phi_k^Z}$ and $\frac{\partial \hat{x}^I[n]}{\partial \phi_k^Z}$ in Eq. (5) are computed as follows:

$$\frac{\partial \hat{x}^R[n]}{\partial \phi_k^R} = \frac{\partial \hat{x}^I[n]}{\partial \phi_k^I} = x^R[M-1+n-k] \tag{6}$$

$$\frac{\partial \hat{x}^I[n]}{\partial \phi_k^R} = -\frac{\partial \hat{x}^R[n]}{\partial \phi_k^I} = x^I[M-1+n-k] \tag{7}$$

The above analysis shows that the relationship between the waveform modification and classification processes can be described by a set of gradients. Most importantly, they can be used to devise effective optimization algorithms that solve Problem (RWOP).

In Section 4.3.2, we design an algorithm to solve Problem (RWOP) and compute the optimal FIR filter parameters $\boldsymbol{\phi}$ by using the Non-linear Conjugate Gradient (NCG) method and the gradients computed in Eq. (6) and Eq. (7). While our simulation results have shown that NCG is more accurate than other gradient-based optimization algorithms (*e.g.*, gradient descent algorithms), we remark that *DeepRadioID* is independent of the actual algorithm used to compute $\boldsymbol{\phi}$, and other approaches can be used to solve Problem (RWOP).

4.3.2 Filter taps computation through NCG. As shown in Figure 2 and discussed in Section 3, *DeepRadioID* iteratively adapts to channel fluctuations by periodically updating the filter taps associated to any given target device A. For the sake of generality, we refer to this periodic update as an *optimization epoch*, and a new epoch is started as soon as one or more *triggering events* are detected by *DeepRadioID*. Triggering events can be either cyclic, *e.g.*, timer timeout, or occasional, *e.g.*, the accuracy for a target devices falls below a minimum desired threshold.

For each epoch i, let $t = 1, 2, \ldots, T$ denote the iteration counter of the optimization algorithm. At each iteration t of the algorithm, the filter taps are updated according to the following iterative rule

$$\boldsymbol{\phi}^{(t)} = \boldsymbol{\phi}^{(t-1)} + \boldsymbol{\alpha}^{(t)} \mathbf{p}^{(\mathbf{t})} \tag{8}$$

In Eq. (8), $\mathbf{p}^{(\mathbf{t})}$ and $\boldsymbol{\alpha}^{(t)}$ represent the search direction and update step of the algorithm, respectively. To put it simple, $\mathbf{p}^{(\mathbf{t})}$ gives us information on the direction to be explored, while $\boldsymbol{\alpha}^{(t)}$ tells us how large the exploration step taken in that direction should be. More in detail, the two terms are computed as follows:

$$\mathbf{p}^{(\mathbf{t})} = \sum_{s=1}^{S} \left(\nabla_{\boldsymbol{\phi}} f_A(\mathbf{x}_s, \boldsymbol{\phi}^{(t-1)}) \right) + \boldsymbol{\beta}^{(t)} \mathbf{p}^{(\mathbf{t-1})} \tag{9}$$

$$\boldsymbol{\alpha}^{(t)} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_{s=1}^{S} f_A(\mathbf{x}_s, \boldsymbol{\phi}^{(t-1)} + \boldsymbol{\alpha} \mathbf{p}^{(\mathbf{t})}) \tag{10}$$

where gradients derive from Eq. (6) and Eq. (7), $\beta^{(1)} = 0$ and $p^{(0)} = 0$. The parameter $\boldsymbol{\beta}^{(t)}$ is defined as

$$\boldsymbol{\beta}^{(t)} = \frac{|| \sum_{s=1}^{S} \left(\nabla_{\boldsymbol{\phi}} f_A(\mathbf{x}_s, \boldsymbol{\phi}^{(t-1)}) \right) ||_2^2}{|| \sum_{s=1}^{S} \left(\nabla_{\boldsymbol{\phi}} f_A(\mathbf{x}_s, \boldsymbol{\phi}^{(t-2)}) \right) ||_2^2} \tag{11}$$

and is generally referred to as the *conjugate gradient* (update) parameter used in NCG methods to improve the space exploration process by speeding up the convergence of the algorithm [7].

Interestingly enough, $\mathbf{p}^{(\mathbf{1})} = \sum_{s=1}^{S} \left(\nabla_{\boldsymbol{\phi}} f_A(\mathbf{x}_s, \boldsymbol{\phi}^{(0)}) \right)$ when $t = 1$. That is, the first iteration of the NCG algorithm corresponds to a classic gradient descent. Also, $\boldsymbol{\alpha}^{(t)}$ in Eq. (10) is computed through line-search algorithms. While both exact and approximated line search algorithms can be considered, there are few aspects that need to be considered when implementing Eq. (10). Indeed, since the function f_A is highly non-linear and has no closed-form representation, to compute Eq. (10) requires the continuous evaluation of $\sum_{s=1}^{S} f_A(\mathbf{x}_s, \boldsymbol{\phi}^{(t-1)} + \boldsymbol{\alpha} \mathbf{p}^{(\mathbf{t})})$ and its first and second order derivatives. For this reason, in some cases it might be computationally expensive to run exact line search algorithms on f_A, and approximated line search algorithms are to be preferred. For example, to speed-up the computation of $\boldsymbol{\alpha}^{(t)}$, we can consider a secant method approximation where the second derivatives of f_A are approximated by using the first order derivatives computed in Eq. (6) and Eq. (7).

5 DeepRadioID FIR COMPENSATION

Although the waveform filtering process is beneficial to the classification process as we can optimally modify the waveform generated by a given target device, it may negatively affect the quality of transmitted data. Indeed, moving I/Q symbols within the complex plane might impact the demodulation and decoding process, thus increasing the BER associated to the received waveform. Furthermore, the expression of the BER in Constraint (C1) is non-linear and possesses exact and/or approximated closed-form representations only in a limited number of cases (*e.g.*, fading channels with known distributions and low-order modulations).

The above discussion shows that to tackle the BER constraint in Problem (WOP) is challenging, since we need to devise mechanisms that are generic enough to be used with any modulation, coding and channel distributions. To overcome the above issues, we observe that our waveform modification relies on Eq. (1), which clearly represents a discrete convolution between the input sequence $\mathbf{x}$ and the filter taps $\boldsymbol{\phi}$. For the sake of illustration, in the following we use the familiar model

$$z[n] = (\mathbf{h} \circledast \hat{\mathbf{x}})[n] + w[n] \tag{12}$$

where each received I/Q symbol $z[n]$ is written as the sum of a noise term $w[n]$ (typically Additive white Gaussian noise) and the n-th element of the discrete convolution $\mathbf{h} \circledast \hat{\mathbf{x}}$ between the channel h and the transmitted filtered sequence $\hat{\mathbf{x}}$. From Eq. (1), we also have that $\hat{\mathbf{x}} = \mathbf{x} \circledast \phi$. Thus, the discrete Fourier transform (DFT) of $\mathbf{z}$ and $\hat{\mathbf{x}}$ can be written as follows:

$$\begin{aligned} Z(\omega) &= H(\omega)\hat{X}(\omega) + W(\omega) \\ &= H(\omega)X(\omega)\Phi(\omega) + W(\omega) \end{aligned} \tag{13}$$

$$X(\omega) = \frac{Z(\omega) - W(\omega)}{H(\omega)\Phi(\omega)} \tag{14}$$

where we have used capital letters to indicate DFTs.

Eq. (14) shows that to reconstruct the original unfiltered I/Q sequence is possible by computing the inverse DFT of each component of the received signal. Furthermore, FIR compensation implies $\text{BER}_A(\mathbf{x}, \boldsymbol{\phi}) = \text{BER}_A(\mathbf{x})$. Since, the optimization variable of Problem (WOP) is $\boldsymbol{\phi}$, Constraint (C1) does not depend on $\boldsymbol{\phi}$ anymore, and thus it can be removed from Problem (WOP). Finally, note that the

above FIR compensation method is independent of the underlying modulation and coding scheme. It means, that FIR compensation is a general approach, and it can be successfully used to tackle Constraint (C1).

Despite the above properties, one might argue that in general it is not possible to compute a perfect estimation of $W(\omega)$ and $H(\omega)$. However, modern wireless networks embed estimation mechanisms that are almost always able to compute fairly accurate estimations $\tilde{H}(\omega)$ and $\tilde{N}(\omega)$, *e.g.*, through training sequences and pilots [17], and the effect of the FIR filter can thus be compensated to a significant extent. To validate this crucial assumption, we ran a number of experiments on our experimental testbed to evaluate the impact of *DeepRadioID*'s FIR filtering on the packet error rate (PER) and throughput (θ) of a wireless transmission.

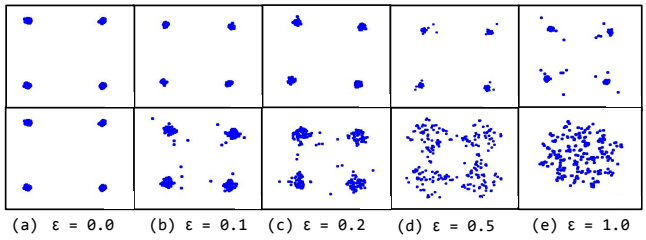

Figure 4: The effect of FIR filtering at the receiver's side for different values of ϵ. Top and bottom sides show respectively the received constellations with/without FIR compensation.

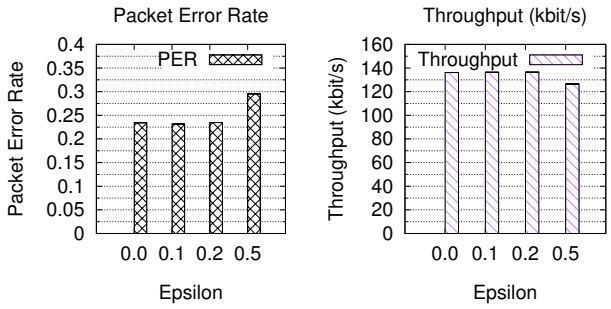

Figure 5: Packet error rate (PER) and Throughput (θ) as a function of ϵ.

Figure 4 shows the received constellations of a QPSK-modulated WiFi transmission where the payload I/Q samples are multiplied in the frequency domain (*i.e.*, FIR filtering) with a random I/Q tap with $I \in [1-\epsilon, 1+\epsilon]$ and $Q \in [0-\epsilon, 0+\epsilon]$). The ϵ parameter represents the relative magnitude of the filter with respect to no filtering, *i.e.*, $\epsilon = 0$. The filtering is then compensated at the receiver's side by using Eq. (14).

Due to the imperfection in channel compensation, we notice than some noise is indeed introduced by the FIR filtering irrespective of our compensation. However, these imperfections do not translate in a significant PER increase. Figure 5 shows the PER and θ as a function of ϵ, which respectively increase and decrease of about 6% and 0.5 kbit/s in the worst case of $\epsilon = 0.5$. However, according to our experiments in Section 6, the ϵ value is typically below 0.2, meaning a PER increase <1% and a θ loss <0.2 kbit/s (0.2%).

6 EXPERIMENTAL RESULTS

In this section, we report the results obtained through extensive experimentation on a practical software-defined radio testbed (Section 6.1), as well as on three datasets of WiFi and ADS-B transmissions obtained through the DARPA RFMLS program (Section 6.4).

6.1 Radio Testbed Setup

Our experimental testbed is composed by twenty software-defined USRP radios acting as transmitters and one USRP acting as receiver. Each USRP has been equipped with a CBX 1200-6000 MHz daughterboard with 40 MHz instantaneous bandwidth [4] and one VERT2450 antennas [3]. Therefore, the RF components of each USRP are nominally-identical. Furthermore, each USRP device sends the same baseband signal , *i.e.*, an IEEE 802.11a/g (WiFi) frame repeated over and over again, to make sure that the deep learning model is learning the hardware impairments and not data patterns.

Figure 6: DeepRadioID Experimental Testbed.

The baseband signal is generated through Gnuradio and then streamed to the selected SDR for over-the-air wireless transmission. The receiver SDR samples the incoming signals at 10 MS/s sampling rate at center frequency of 2.432 GHz. The collected baseband signal is then channel-equalized using IEEE 802.11 pilots and training sequences [17]. Next, the payload I/Q samples are extracted and partitioned into a *sample*. In our experiments, we fix the sample length to $48 \cdot 6 = 288$ I/Q values, corresponding to 6 OFDM symbols containing 48 payload I/Q values. Each of these samples are then used for training and classification.

6.2 Deep Learning Architecture

We use the CNN architecture reported in [15] and depicted in Figure 7. Specifically, each I/Q input sequence is represented as a two-dimensional real-valued tensor of size 2×288. This is then fed to the first convolutional layer (ConvLayer), which consists of 50 filters each of size 1×7. Each filter learns a 7-sample variation in time over the I or Q dimension separately, to generate 50 distinct feature maps over the complete input sample. Similarly, the second ConvLayer has 50 filters each of size 2×7. Each ConvLayer is followed by a Rectified Linear Unit (ReLU) activation and a maximum pooling (MaxPool) layer with filters of size 2×2 and stride 1, to perform a pre-determined non-linear transformation on each element of the convolved output.

The output of the second convolution layer is then provided as input to the first fully connected layer, which has 256 neurons. A second fully connected layer of 80 neurons is added to extract higher level non-linear combinations of the features extracted from previous layers, which are finally passed to a classifier layer. To overcome overfitting, we set the dropout rate to 50% at the dense

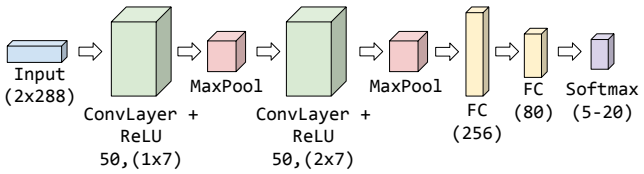

Figure 7: CNN used in our radio testbed experiments.

layers. A *softmax* classifier is used in the last layer to output the probabilities of each sample being fed to the CNN.

We use an ℓ_2 regularization parameter $\lambda = 0.0001$. The weights of the network are trained using Adam optimizer with a learning rate of $l = 0.0001$. We minimize the prediction error through back-propagation, using categorical cross-entropy as a loss function computed on the classifier output. We implement our CNN architecture in Keras running on top of TensorFlow on a system with 8 NVIDIA Cuda enabled Tesla K80m GPU. To train our CNN, we constructed a dataset by performing 10 different transmissions of length 5 minutes for each of the 20 devices, each transmission spaced approximately 5 minutes in time.

6.2.1 Performance Metrics. Hereafter, we will use the following two metrics to assess the performance of our learning system:

(1) *Per-Slice Accuracy (PSA).* As in Section 4.3, a *slice* is a set of S consecutive input. The PSA is thus defined as the average CNN fingerprinting accuracy on the S-sample slice. Since the FIR filter optimized by *DeepRadioID* is computed on one slice, the PSA measures how much *DeepRadioID* is able to increase the short-term accuracy of the CNN.
(2) *Per-Batch Accuracy (PBA).* We define as *batch* as a set of B consecutive slices. The PBA is thus defined as the average CNN fingerprinting accuracy on the B-slice batch. The PBA measures the impact of the optimal FIR filter on the long-term accuracy of the CNN.

In the following experiments, we use live-collected data (*i.e.*, not coming from the training dataset) to compute the PSA and PBA. This data was collected 7 days after the data used for training the dataset was collected. To allow experiments' repeatability, we record a transmission from a given device for about one minute. Then, we select the first slice from the recording and compute the PSA with no FIR optimization. Next, we perform FIR optimization on the slice, re-filter each sample in the slice, and compute the PSA after FIR filtering. To compute the PBA, we apply the same FIR filter to the $B \cdot N - 1$ slices collected after the first one and then compute the average PSA on the B-slice batch.

6.3 DeepRadioID Testbed Results

Figure 8 shows the average PSA and PBA obtained by optimizing three devices over ten different recordings PSA as a function of the model size (*i.e.*, 5 to 20 devices). We also show 95% confidence intervals.

The results obtained in Figure 8 show that *DeepRadioID* is significantly effective in improving the PSA and PBA of our CNN-based fingerprinting system, which are improved by an average of about 57% and 35%, respectively. By accounting that the metrics

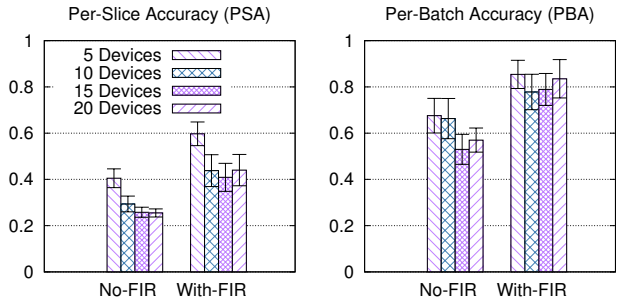

Figure 8: PSA and PBA, Experimental Testbed.

are computed on live-collected data and on a testbed of nominally-identically radios, we consider these results remarkable.

Figure 9 shows the PSA and the PBA obtained by an adversary trying to imitate another device's fingerprint by applying the same FIR. In these experiments, we fixed one device as adversary and used the FIR from other three devices to compute its PSA and PBA for each of its ten recordings. Figure 9 shows that by using the legitimate device's FIR, the adversary transitions from an average PSA of 12% to an average PSA of about 6% (50% decrease), corresponding to an average PBA of about 4%. This ultimately confirms that since a FIR is optimized for a device's specific channel and impairments, an adversary cannot use a legitimate device's FIR to imitate its fingerprint.

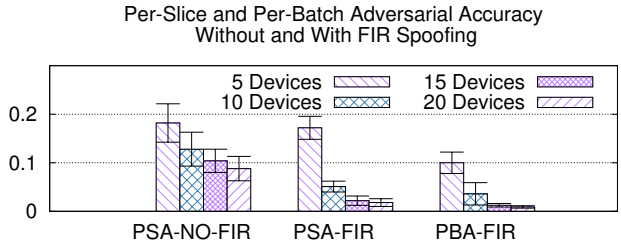

Figure 9: Adversarial Action, Experimental Testbed.

6.4 DeepRadioID Dataset Results

To investigate *DeepRadioID*'s performance on large-scale wireless systems and with different wireless standards, we consider (i) a 500-device dataset of IEEE 802.11a/g (WiFi) transmissions; and (ii) a 500-airplane dataset of Automatic Dependent Surveillance – Broadcast (ADS-B) beacons, both obtained through the DARPA RFMLS program. ADS-B is a surveillance transmission where an aircraft determines its position via satellite navigation and periodically broadcasts it at center frequency 1.090 GHz with sampling rate 1 MSPS and pulse position modulation. This makes ADS-B ideal to evaluate *DeepRadioID*'s performance on different channel/modulation scenarios. For both the WiFi and ADS-B datasets, data collection was performed "in the wild" (*i.e.*, no controlled environment) with a Tektronix RSA operating at 200 MSPS. For the WiFi dataset, as in Section 6.3 we demodulated the transmissions and trained our models on the derived I/Q samples. To demonstrate

the generality of *DeepRadioID*, the ADSB model was instead trained on the unprocessed I/Q samples.

To handle the increased number of devices and experiment with a different CNN, we use the CNN architecture shown in Figure 10. In the case of ADS-B we train our CNN on examples containing 1024 consecutive unprocessed I/Q samples. For WiFi, we only use 128 samples. Unless stated otherwise, PSA and PBA are computed on batches of 12 slices, each containing 25 inputs.

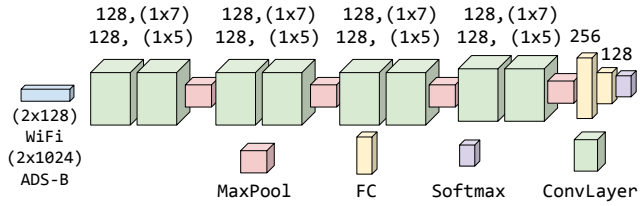

Figure 10: CNN used in our dataset experiments.

To compare our approach with the state of the art by Vo *et al.* [18], which reports results on 93 devices, we trained our model on a subset of 100 devices, achieving PSA and PBA of .32 and .44, respectively. Figure 11 shows the PSA and PBA improvement brought by *DeepRadioID* FIR filtering as function of the number of FIR taps. In particular, the PBA improvement is around 30% when 10 FIR taps are used, which brings the accuracy to about 74% on the average, outperforming [18] which is 47%.

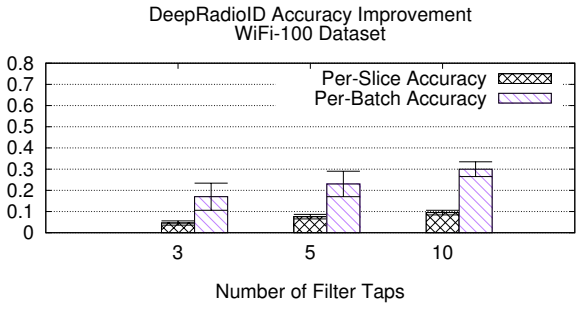

Figure 11: PSA/PBA Improvement, WiFi-100.

Figure 12 reports the PSA and PBA improvement on the ADS-B-500 dataset (baseline PSA and PBA of .5028 and .6193), which reach respectively about 10% and 50% in the case of 10 filter taps. Figure 12 also shows that, although the ADSB-500 PSA improvement is about the same experienced in the WiFi-100 model, the PBA improvement is significantly higher in ADSB-500 (30% vs 50%). This is thanks to the fact that a very small increase in PSA usually corresponds to a significant increase in PBA when the number of devices is higher. Also, ADS-B transmits in a less-crowded channel than WiFi, thus we expect our model and *DeepRadioID* to perform better on ADSB-500 when the number of inputs per slice increases.

This is also confirmed by Figure 13, where we show the PSA and PBA improvement in the WiFi-500 dataset as a function of the number of FIR taps and the number of inputs in each slice (25 and 100). Indeed, Figure 13 shows a similar PBA improvement when 10 FIR taps are considered. Furthermore, Figure 13 shows

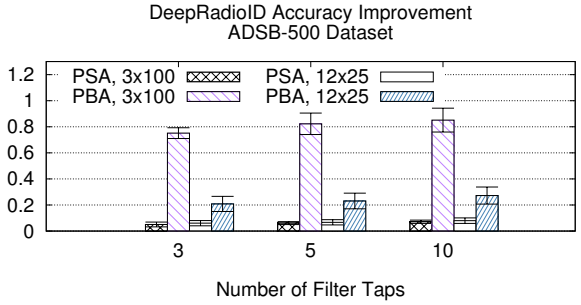

Figure 12: Per-Slice and Per-Batch Accuracy Improvement, ADSB-500, as a function of the number of inputs per slice (100 and 25) and the number of batches (3 and 12).

that the number of inputs per slice significantly impacts on the fingerprinting accuracy, especially on the PBA, in both ADSB-500 and WiFi-500. This is because (i) the FIR optimization increases in effectiveness as the number of inputs per slice increases, because the FIR is averaged over more channel realizations; and (ii) the boosting effect given by the PBA increases as the number of slices increases.

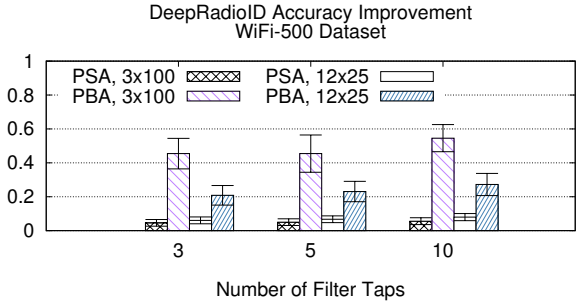

Figure 13: Per-Slice and Per-Batch Accuracy Improvement, WiFi-500, as a function of the number of inputs per slice (100 and 25) and the number of batches (3 and 12).

Figure 14 shows the PSA improvement for ADSB-500 and WiFi-500 as a function of the number of FIR taps. As we can see, the PSA improvement converges to approximately .15 as more FIR taps are used, in both cases. Very interestingly, Figure 14 also shows that WiFi-500 converges more rapidly than ADSB-500. This is due to the fact that our ADSB-500 model was trained on unprocessed I/Q samples (*i.e.*, without any channel equalization). Therefore, the number of FIR taps than *DeepRadioID* needs to obtain the same performance as in WiFi-500 increases, as the FIR has to compensate for a significant channel action already mitigated in WiFi-500.

Finally, Figure 15 depicts the average FIR ϵ-value (defined in Section 5) as a function of the number of taps and the model considered. As anticipated in Section 5, Figure 15 concludes that the maximum ϵ-value is below .2, which allows us to achieve low BER even when the FIR is applied. Interestingly enough, we notice that ADSB-500 requires taps that are on average higher than WiFi-100 and WiFi-500. This is due to the fact that ADSB-500 operates on

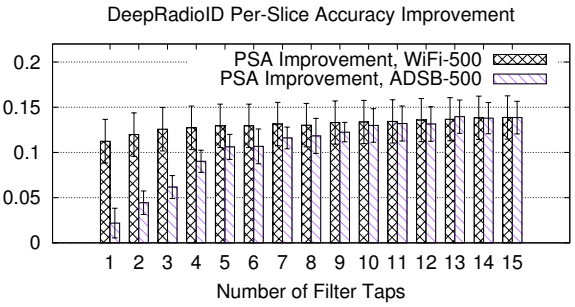

Figure 14: PSA Improvement as a function of the number of FIR taps. ADSB-500 converges slower than WiFi-500 given the FIR has to operate on unprocessed I/Q samples.

unprocessed I/Q samples in the time domain, therefore the taps required to modify the signal need to be greater.

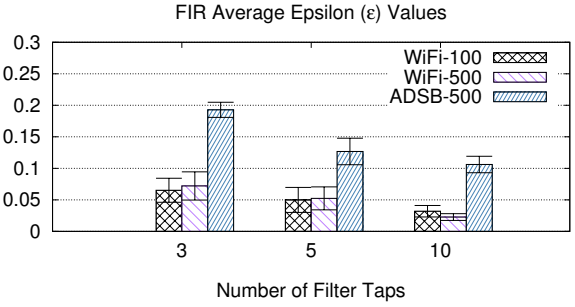

Figure 15: Average FIR Epsilon (ϵ) values as a function of the number of FIR taps.

7 CONCLUSIONS

In this paper we have proposed *DeepRadioID*, a system to optimize the accuracy of deep-learning-based radio fingerprinting algorithms. We have extensively evaluated *DeepRadioID* on a experimental testbed of 20 nominally-identical software-defined radios, as well as on datasets made up by WiFi and ADS-B transmissions. Experimental results have shown that *DeepRadioID* (i) increases fingerprinting accuracy by about 35%, 50% and 58% on the three scenarios considered; (ii) decreases an adversary's accuracy by about 54% when trying to imitate other device's fingerprints by using their filters; (iii) achieves 27% improvement over the state of the art on a 100-device dataset.

ACKNOWLEDGMENT

This work is supported by the Defense Advanced Research Projects Agengy (DARPA) under RFMLS program contract N00164-18-R-WQ80. We are sincerely grateful to Paul Tilghman, program manager at DARPA, Esko Jaska, our shepherd Srinivas Shakkottai and the anonymous reviewers for their insightful comments and suggestions. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

REFERENCES

[1] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh. 2008. Wireless device identification with radiometric signatures. In *Proceedings of the 14th ACM international conference on Mobile Computing and Networking (MobiCom)*. ACM, 116–127.

[2] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. 2014. cuDNN: Efficient Primitives for Deep Learning. *arXiv preprint arXiv:1410.0759* (2014).

[3] Ettus Research (A National Instrument Company). 2016. VERT2450 Antenna. https://www.ettus.com/product/details/VERT2450.

[4] Ettus Research (A National Instrument Company). 2018. CBX 1200-6000 MHz Rx/Tx (40 MHz). https://www.ettus.com/product/details/CBX.

[5] Aidin Ferdowsi and Walid Saad. 2018. Deep learning-based dynamic watermarking for secure signal authentication in the Internet of Things. In *2018 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[6] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT press Cambridge.

[7] Magnus Rudolph Hestenes and Eduard Stiefel. 1952. *Methods of conjugate gradients for solving linear systems*. Vol. 49. NBS Washington, DC.

[8] Jithin Jagannath, Nicholas Polosky, Anu Jagannath, Francesco Restuccia, and Tommaso Melodia. 2019. Machine Learning for Wireless Communications in the Internet of Things: A Comprehensive Survey. *arXiv preprint arXiv:1901.07947* (2019).

[9] E. Johnson. 1966. Physical limitations on frequency and power parameters of transistors. In *1958 IRE International Convention Record*, Vol. 13. IEEE, 27–34.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436.

[11] Nam Tuan Nguyen, Guanbo Zheng, Zhu Han, and Rong Zheng. 2011. Device fingerprinting to enhance wireless security using nonparametric Bayesian method. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 1404–1412.

[12] Linning Peng, Aiqun Hu, Junqing Zhang, Yu Jiang, Jiabao Yu, and Yan Yan. 2019. Design of a Hybrid RF Fingerprint Extraction and Device Classification Scheme. *IEEE Internet of Things Journal* 6, 1 (Feb 2019), 349–360. https://doi.org/10.1109/JIOT.2018.2838071

[13] Francesco Restuccia, Salvatore D'Oro, and Tommaso Melodia. 2018. Securing the Internet of Things in the Age of Machine Learning and Software-Defined Networking. *IEEE Internet of Things Journal* 5, 6 (Dec 2018), 4829–4842. https://doi.org/10.1109/JIOT.2018.2846040

[14] Francesco Restuccia and Tommaso Melodia. 2019. Big Data Goes Small: Real-time Spectrum-Driven Embedded Wireless Networking through Deep Learning in the RF Loop. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*.

[15] Shamnaz Riyaz, Kunal Sankhe, Stratis Ioannidis, and Kaushik Chowdhury. 2018. Deep Learning Convolutional Neural Networks for Radio Identification. *IEEE Communications Magazine* 56, 9 (Sept 2018), 146–152. https://doi.org/10.1109/MCOM.2018.1800153

[16] Sabrina Sicari, Alessandra Rizzardi, Luigi Alfredo Grieco, and Alberto Coen-Porisini. 2015. Security, Privacy and Trust in Internet of Things: The Road Ahead. *Computer Networks* 76 (2015), 146–164.

[17] Essam Sourour, Hussein El-Ghoroury, and Dale McNeill. 2004. Frequency Offset Estimation and Correction in the IEEE 802.11 a WLAN. In *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, Vol. 7. IEEE, 4923–4927.

[18] Tien Dang Vo-Huu, Triet Dang Vo-Huu, and Guevara Noubir. 2016. Fingerprinting Wi-Fi devices using software defined radios. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 3–14.

[19] Feiyi Xie, Hong Wen, Yushan Li, Songlin Chen, Lin Hu, Yi Chen, and Huanhuan Song. 2018. Optimized Coherent Integration-Based Radio Frequency Fingerprinting in Internet of Things. *IEEE Internet of Things Journal* 5, 5 (Oct 2018), 3967–3977. https://doi.org/10.1109/JIOT.2018.2871873

[20] Yuexiu Xing, Aiqun Hu, Junqing Zhang, Linning Peng, and Guyue Li. 2018. On Radio Frequency Fingerprint Identification for DSSS Systems in Low SNR Scenarios. *IEEE Communications Letters* 22, 11 (Nov 2018), 2326–2329. https://doi.org/10.1109/LCOMM.2018.2871454

[21] Qiang Xu, Rong Zheng, Walid Saad, and Zhu Han. 2016. Device fingerprinting in wireless networks: Challenges and opportunities. *IEEE Communications Surveys & Tutorials* 18, 1 (2016), 94–104.

[22] Kai Zhao and Lina Ge. 2013. A survey on the internet of things security. In *Computational Intelligence and Security (CIS), 2013 9th International Conference on*. IEEE, 663–667.

[23] Michele Zorzi, Alexander Gluhak, Sebastian Lange, and Alessandro Bassi. 2010. From today's Intranet of Things to a future Internet of Things: a Wireless-and Mobility-related View. *IEEE Wireless communications* 17, 6 (2010).