DeepFIR: Channel-Robust Physical-Layer Deep Learning Through Adaptive Waveform Filtering

Francesco Restuccia, *Member, IEEE*, Salvatore D'Oro, *Member, IEEE*, Amani Al-Shawabka,
Bruno Costa Rendon, Stratis Ioannidis, and Tommaso Melodia, *Fellow, IEEE*

Abstract—Deep learning can be used to classify waveform characteristics (*e.g.*, modulation) with accuracy levels that are hardly attainable with traditional techniques. Recent research has demonstrated that one of the most crucial challenges in wireless deep learning is to counteract the channel action, which may significantly alter the waveform features. The problem is further exacerbated by the fact that deep learning algorithms are hardly re-trainable in real time due to their sheer size. This paper proposes *DeepFIR*, a framework to counteract the channel action in wireless deep learning 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 features according to the current channel conditions. We mathematically formulate the *Waveform Optimization Problem* (WOP) as the problem of finding the optimum FIR to be used on a waveform to improve the classifier's accuracy. We also propose a data-driven methodology to train the FIRs directly with dataset inputs. We extensively evaluate *DeepFIR* on an experimental testbed of 20 software-defined radios, as well as on two datasets made up by 500 ADS-B devices and by 500 WiFi devices and a 24-class modulation dataset. Experimental results show that our approach (i) increases the accuracy of the radio fingerprinting models by about 35%, 50% and 58%; (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; (iv) increases by 2x the accuracy of the modulation dataset.

Index Terms—Deep learning, wireless, security, testbed, software-defined radio, datasets.

I. INTRODUCTION

THE sheer number of wireless devices that will soon join the Internet of Things (IoT) will inevitably make the radio spectrum an extremely dynamic environment, where closed-form models will be difficult or impossible to formulate [1]. To address the severely increased spectrum
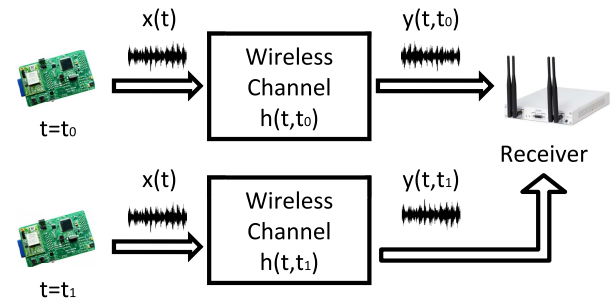
Fig. 1.   The channel problem in wireless deep learning.

complexity, the community is using data-driven *deep learning* techniques directly spun off of the computer vision domain [2].

Deep learning allows to capture relevant features in classification problems lacking tractable closed-form mathematical formulations. These models are based on highly non-linear neural networks and can address wireless problems such as signal/traffic classification [3]–[5], radio fingerprinting [6], [7] and resource allocation [8], among others [9].

Although wireless deep learning has proven to be effective, recent research [10], [11] has demonstrated that the wireless channel can seriously compromise the effectiveness of the features learned by the classifier. The key issue in wireless deep learning is that classifiers operating on waveforms are by definition *time-varying* systems due to the ever-changing nature of the channel [12]. Figure 1 shows a (simplified) representation of the channel issue in wireless deep learning.

Addressing time variance is one of modern machine learning's greatest issues [13], which makes the channel problem in wireless deep learning significantly challenging. One possible option is to periodically retrain the model; however, deep learning networks usually require a significant time to be retrained, even with modern GPUs [14]. Therefore, we cannot assume that *the underlying deep learning model can be retrained in real time* before the channel has changed the features again. Another key reason to avoid the re-training of the model is the additional energetic burden imposed to the overall wireless infrastructure, which has to continuously (i) collect new waveforms, (ii) fine-tune the model with the new data, and (iii) re-send the new weights to the wireless platform. Thus, this approach is highly inefficient – and ultimately ineffective since it is hardly scalable to IoT networks.

In other words, a waveform $x(t)$ transmitted at time t_0 and t_1 by the same device will necessarily be received with different levels of distortion by the receiver. Since the training process cannot include all the possible channel realizations and distributions, the features learned by the classifier cannot be entirely time-invariant [6], [7], [15]. The seriousness of this problem has been recently exposed in the context of radio fingerprinting [11], where the authors have demonstrated that the accuracy of the classifier can drop down to almost zero when tested on I/Q samples collected in days different than the one when samples were collected for training.

To address this spinous issue, in our preliminary work [10] we have introduced *DeepRadioID*, a framework for channel-resilient optimization of deep learning-based radio fingerprinting algorithms. The core intuition was to impose a carefully-tailored modification of the waveform at the transmitter's side to "restore" the features compromised by the channel. To achieve this goal, the transmitter applies a digital finite input response filter (FIR) to the waveform before transmission, which slightly modifies its baseband signal to compensate for current channel conditions. The receiver is tasked with computing the FIR through mathematical optimization and then send it back to the transmitter as feedback, which will use the FIR to filter its own I/Q samples. We also demonstrated that an adversary cannot reuse the FIR computed for another device to increase the probability to be identified as that device.

This paper builds upon and extends our work [10] by making the following novel contributions:

- We propose *DeepFIR*, a real-time channel- and adversary-resilient optimization system to improve the accuracy of a *generic* wireless deep learning model. 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 V 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). We further propose a data-driven approach implemented in Keras to compute the optimal FIR without the need of external optimization algorithms (Section VI);

- We extensively evaluate the performance of *DeepFIR* 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). Testbed results (Section VII) indicate that (i) an adversary trying to imitate a fingerprint by using the same FIR filter *decreases* its fingerprinting accuracy by about 54% on the average; (ii) DeepFIR *increases* the accuracy by 35% on the average;

- To further evaluate the scalability of *DeepFIR* and to experiment with different wireless technologies, deeper learning models and learning problems we also leverage (i) two datasets of IEEE 802.11a/g (WiFi) and Automatic Dependent Surveillance – Broadcast (ADS-B)

transmissions, each containing 500 devices[1] (Section VIII); and (ii) on a widely-available modulation recognition model [3] trained on the RadioML 2018.01A dataset (Section IX), which includes 24 different analog and digital modulations with different levels of signal-to-noise ratio (SNR);

- Results on the large-scale datasets show that *DeepFIR* increases the accuracy by 50% and by 58% on the 500-device ADS-B and WiFi datasets, respectively, and by comparing with the state of the art [16], *DeepFIR* improves the accuracy by about 27% on a reduced dataset of 100 WiFi devices. Finally, our data-driven *DeepFIR* methodology improves the accuracy of the modulation classifier of $2\times$ on the average.

II. RELATED WORK

Deep learning at the physical layer (PHY) of the wireless protocol stack has received significant attention over the last few years [17]. Most of existing work focused on developing techniques based on machine learning (ML) to fingerprint wireless devices. One of the first papers to address radio fingerprinting is due to Nguyen *et al.* [18], who used device-dependent channel-invariant radio-metrics and propose a non-parametric Bayesian method to detect the number of devices. Recently, Vo-Huu *et al.* [16] 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. Peng *et al.* [19] proposed fingerprinting algorithms for Zig-Bee 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 usage of supervised machine learning techniques, moreover, has received a lot of attention over the last years, particularly the problem of modulation recognition [20]–[22]. Similar to radio fingerprinting, feature-based ML constitutes the majority of existing work [23]–[27].

The key issue with legacy ML techniques is its inherent feature extraction process, which involves the computation of waveform characteristics such as kurtosis, high-order cyclic moments, median, average and so on. Feature extraction is also very application-specific in nature, which hinders flexibility and adaptation. Thus, its applicability for real-time wireless spectrum analysis presents substantial limitations. Recently, deep learning [2] has been proposed as a viable solution [28] to perform complex signal classification without incurring in manual feature extraction. Among others, deep learning has been extensively used to address modulation recognition [3], [29]–[33]. For example, O'Shea *et al.* proposed in [3] several modulation classifiers, while in [33] Szegedy *et al.* used deep neural networks to classify modulation class and order. Portion of the above work did consider the issue of signal-to-noise (SNR) ratio and other waveform impairments when evaluating the model performance. However, it did not propose any

[1] Due to contract obligations, we cannot release the datasets to the community. We hope this will change in the future.

systematic investigation of the role of the wireless channel in the overall performance.

To the best of our knowledge, Al-Shawabka *et al.* [11] was the first work to determining the impact of the wireless channel on the performance of deep learning algorithms, by definitely establishing through extensive performance evaluation that the time-varying nature of the channel impacts significantly the classification accuracy. In our previous work [10], we proposed *DeepRadioID*, a framework to improve the channel resilience of deep learning-based radio fingerprinting algorithms through the application of carefully-optimized finite response input (FIR) filters. This paper substantially extends our prior work by proving that our FIR-based approach is applicable to address general deep learning models and not only radio fingerprinting models, by appling *DeepFIR* to improve the performance of a modulation classifier proposed in [3]. Moreover, we have designed and developed a Keras-based version of our system, which is able to run on GPU and thus reduce significantly the FIR computation time.

III. DEEPFIR: AN OVERVIEW

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

A. DeepFIR: Key Intuitions

The need to optimize the accuracy of wireless deep learning systems arises from the fact that the wireless channel is dynamic and almost unpredictable in nature. Moreover, hardware impairments such as I/Q imbalances, DC offset, phase noise, carrier/sampling offsets, and power amplifier distortions are extremely time-varying and dependent on a number of factors, such as local oscillator (LO) frequency [34] and current temperature of the RF circuitry [35]. These considerations imply that we cannot assume waveforms as perfectly stationary – hence the need for real-time optimization.

To address the complex 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 [2] – 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 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 $\mathcal{O}(m)$ – 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 V.
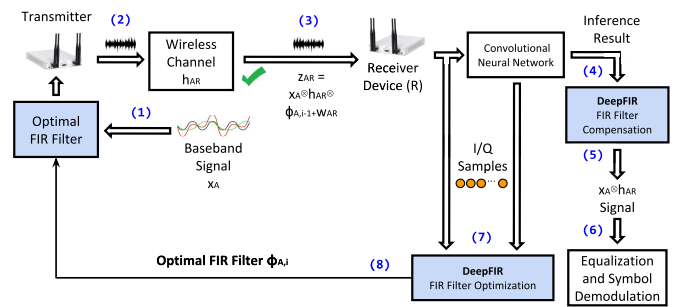

Fig. 2. A high-level overview of the *DeepFIR* system. Since A's FIR filter has been tailored to match A's unique channel conditions, we show in Section VII that an adversary cannot leverage A's filter. Notice that the transmitter does not rely on the classifier to apply its FIR filter. Instead, the classifier is entirely located at the receiver's side.

However, this approach spurs another challenge, which is *how to set the FIR taps in such a way that the classification accuracy for a given class 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 class, as shown in Section IV-C. 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.

B. DeepFIR: A Walk-Through

Figure 2 provides a walk-through of the main building blocks of *DeepFIR* and the main operations involved in the classification process. We highlight with a shade of blue the blocks that are added to the normal modulation/demodulation chain as part of *DeepFIR*. The walk-through also shows how an adversary may try to camouflage its waveform of being of a different class (e.g., imitate the fingerpritint of another device). The detailed explanation of *DeepFIR*'s main module will be given in Section IV.

The first step for a device "A" that transmits a waveform to a receiver "R" is to filter its baseband signal with FIR $\phi_{A,i-1}$ (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 $\mathbf{z}_{AR} = \mathbf{x}_A \circledast \phi_{A,i-1} \circledast \mathbf{h}_{AR} + \mathbf{w}_{AR}$, 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 classify the waveform (step 4). The fingerprinting result is then used to compensate the FIR filter $\phi_{A,i-1}$ (step 5, discussed in Section V), 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 classification result are then fed to the *DeepFIR* FIR Optimization module (step 7, presented in Section IV). The optimal FIR filter $\phi_{A,i}$ is then sent back to A (step 8).

IV. DeepFIR FIR Optimization

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

A. Background Notions and Definitions

Let us define as *input* a set of N consecutive I/Q samples. 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 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 $(f_1, f_2, \ldots, f_D) \in \mathcal{Y}$, where the i-th component denote the probability that the input fed to the CNN belongs to device i.

DeepFIR relies on discrete finite impulse response filters (in short, FIRs) to achieve adaptive waveform modification. FIRs have the advantage that 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 ϕ of M *filter taps*, i.e., $\phi = (\phi_1, \phi_2, \ldots, \phi_M)$. 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 the wireless channel operates 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:

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

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 $k > 1$.

B. 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
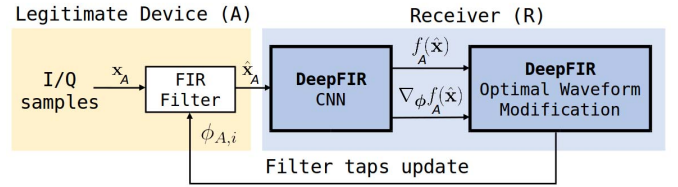

Fig. 3.   Waveform modification optimization loop.

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 accuracy for a given class.

Figure 3 shows a block diagram of the waveform modification optimization loop performed by *DeepFIR*. 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 class for which we want to improve the detection accuracy of the CNN, and let $\phi_{A,i}$ be the filter taps associated to the target class 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 $\phi_{A,i}$. Specifically, we have that

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

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

C. Waveform Optimization Problem (WOP)

We can now formally define the objective of *DeepFIR* as follows: (i) maximize the accuracy of the classifier for a specific target class A; and (ii) to guarantee that the resulting BER does not exceed a given maximum tolerable threshold $\mathrm{BER}_{\max}$. Since we aim at achieving channel-resilient waveform modification, we need to compute a FIR parameter configuration ϕ_A that can be used for multiple consecutive transmissions. To compute different ϕ_A values for each single input $\mathbf{x}$ is inefficient in many cases, *i.e.*, if applied to another input sequence $\mathbf{x}' \neq \mathbf{x}$, the FIR might decrease the accuracy of the classifier.

To overcome the above problem, rather than maximizing the accuracy of the classifier on an input-by-input basis, we compute the FIR ϕ_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:

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

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

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. 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, 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 V, the impact of the waveform modification procedure on the BER of communications among the receiver and the target device A is negligible. Indeed, as shown in Figure 2, *DeepFIR* 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

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

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

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

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

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

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

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:

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

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

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 IV-C.2, we design an algorithm to solve Problem (RWOP) and compute the optimal FIR filter parameters $\boldsymbol{\phi}$ by using the Nonlinear 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 *DeepFIR* is independent of the actual algorithm used to compute $\boldsymbol{\phi}$, and other approaches can be used to solve Problem (RWOP).

2) Filter Taps Computation Through NCG: As shown in Figure 2 and discussed in Section III, *DeepFIR* 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 *DeepFIR*. 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: $\boldsymbol{\phi}^{(t)} = \boldsymbol{\phi}^{(t-1)} + \boldsymbol{\alpha}^{(t)} \mathbf{p^{(t)}}$, where $\mathbf{p^{(t)}}$ and $\boldsymbol{\alpha}^{(t)}$ represent the search direction and update step of the algorithm, respectively. To put it simple, $\mathbf{p^{(t)}}$ 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:

$$\mathbf{p}^{(\mathbf{t})} = \sum_{s=1}^{S} \left(\nabla_{\phi} f_A(\mathbf{x}_s, \phi^{(t-1)}) \right) + \boldsymbol{\beta}^{(t)} \mathbf{p}^{(\mathbf{t-1})} \qquad (8)$$

$$\boldsymbol{\alpha}^{(t)} = \mathrm{argmax}_{\boldsymbol{\alpha}} \sum_{s=1}^{S} f_A(\mathbf{x}_s, \phi^{(t-1)} + \boldsymbol{\alpha}\mathbf{p}^{(\mathbf{t})}) \qquad (9)$$

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

$$\beta^{(t)} = \frac{\| \sum_{s=1}^{S} \left(\nabla_{\phi} f_A(\mathbf{x}_s, \phi^{(t-1)}) \right) \|_2^2}{\| \sum_{s=1}^{S} \left(\nabla_{\phi} f_A(\mathbf{x}_s, \phi^{(t-2)}) \right) \|_2^2} \qquad (10)$$

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 [36].

Algorithm 1 Nonlinear Conjugate Gradient (NCG) to Solve Problem (RWOP)

1: Input Target class $A \in \{1, \ldots, D\}$, Input slices $(\mathbf{x}_{\mathbf{s}=1,\ldots,S})$, ϵ;
2: Output A FIR filter ϕ;
3: Initialize $\boldsymbol{\beta}^{(1)} = 0$, $\mathbf{p}^{(0)} = 0$,
4: $t \leftarrow 1$;
5: $\phi(t-1) \leftarrow$ random taps initialization with parameter ϵ;
6: $\phi(t-2) \leftarrow \phi(t-1)$;
7: **while** Converge criterion not satisfied **do**
8:     $\beta^{(t)} \leftarrow \frac{\| \sum_{s=1}^{S} (\nabla_{\phi} f_A(\mathbf{x}_s, \phi^{(t-1)})) \|_2^2}{\| \sum_{s=1}^{S} (\nabla_{\phi} f_A(\mathbf{x}_s, \phi^{(t-2)})) \|_2^2}$;
9:     $\mathbf{p}^{(t)} \leftarrow \sum_{s=1}^{S} \left(\nabla_{\phi} f_A(\mathbf{x}_s, \phi^{(t-1)}) \right) + \boldsymbol{\beta}^{(t)} \mathbf{p}^{(t-1)}$;
10:    $\boldsymbol{\alpha}^{(t)} \leftarrow \mathrm{argmax}_{\boldsymbol{\alpha}} \sum_{s=1}^{S} f_A(\mathbf{x}_s, \phi^{(t-1)} + \boldsymbol{\alpha}\mathbf{p}^{(t)})$;
11:    $\boldsymbol{\phi}^{(t)} \leftarrow \phi^{(t-1)} + \boldsymbol{\alpha}^{(t)} \mathbf{p}^{(t)}$
12:    $t \leftarrow t + 1$;
13: **end while**
14: Return $\boldsymbol{\phi}^{(t)}$

Interestingly enough, $\mathbf{p}^{(\mathbf{1})} = \sum_{s=1}^{S} \left(\nabla_{\phi} f_A(\mathbf{x}_s, \phi^{(0)}) \right)$, meaning that when $t = 1$ NCG is a classic gradient descent. Also, $\boldsymbol{\alpha}^{(t)}$ in Eq. (9) 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. (9). Indeed, since the function f_A is highly non-linear and has no closed-form representation, to compute Eq. (9) requires the continuous evaluation of $\sum_{s=1}^{S} f_A(\mathbf{x}_s, \phi^{(t-1)} + \boldsymbol{\alpha}\mathbf{p}^{(\mathbf{t})})$ and its first and second order derivatives. For this reason, 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 where the second derivatives of f_A are approximated by using the first order derivatives computed in Eq. (6) and Eq. (7).

V. DEEPFIR 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. 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). 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 ϕ. For the sake of illustration, in the following we use the familiar model

$$z[n] = (\mathbf{h} \circledast \hat{\mathbf{x}})[n] + w[n] \qquad (11)$$

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:

$$Z(\omega) = H(\omega)\hat{X}(\omega) + W(\omega)$$
$$= H(\omega)X(\omega)\Phi(\omega) + W(\omega) \qquad (12)$$
$$X(\omega) = \frac{Z(\omega) - W(\omega)}{H(\omega)\Phi(\omega)} \qquad (13)$$

where we have used capital letters to indicate DFTs.

Eq. (13) 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 $\mathrm{BER}_A(\mathbf{x}, \phi) = \mathrm{BER}_A(\mathbf{x})$. Since, the optimization variable of Problem (WOP) is ϕ, Constraint (C1) does not depend on ϕ 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. This 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 [34], 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 *DeepFIR*'s FIR filtering on the packet error rate (PER) and throughput (θ) of a wireless transmission.

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. (13).

We notice than some noise is indeed introduced by the FIR filtering irrespective of our compensation. However, Figure 5 shows the PER and θ as a function of ϵ, which respectively
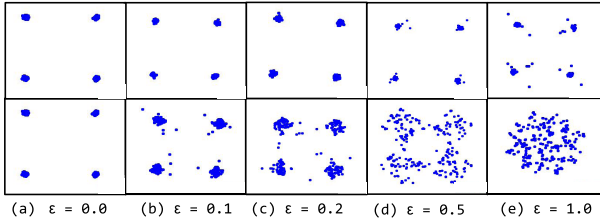
Fig. 4. The effect of FIR filtering at the receiver's side for different values of ϵ. Top and bottom sides show respectively the received constellations with/without FIR compensation.
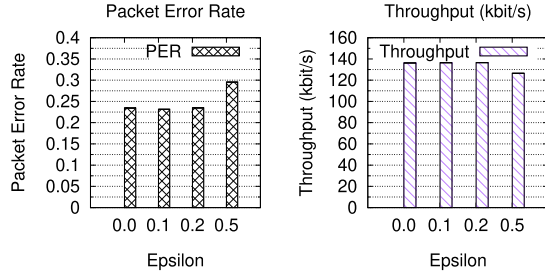

Fig. 5. Packet error rate (PER) and Throughput (θ) as a function of ϵ.

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 VII, the ϵ value is typically below 0.2, meaning a PER increase $<1\%$ and a θ loss <0.2 kbit/s (0.2%).

VI. Data-Driven FIR Optimization

Although traditional optimization techniques can indeed be leveraged to compute FIR taps for each individual class, these techniques algorithms usually suffer from scalability issues, primarily due to the complexity of computing an optimal solution. Although this might not be an issue when the number of classes and/or the size of the dataset are small, the same does not hold anymore when both the problem and the inputs grow in size. Also, while trained models leverage well-established machine learning software libraries (such as Keras, TensorFlow and PyTorch) that take advantage of GPU processing to speed-up computation, the majority of optimization numerical solvers (*e.g.*, Matlab, CPLEX) run on CPUs and do not directly interface with neural network models. A direct consequence of this latter aspect is that the numerical solver and the training model run on separate environments, which inevitably introduces computation delays especially when computing gradients. To overcome the above limitations, in this section we present a novel data-driven approach to compute FIR taps for each class individually. The code repository is available for download at https://github.com/frestuc/DeepFIR.

Equation (2) shows that FIRs can be modeled as a discrete convolution between FIR taps and the input data. Thus, the same operation can be implemented as an *FIR Layer*, whose weights represent FIR taps and whose output follows (2). Since each class requires an individual FIR tap configuration, we can generate D distinct FIR Layers whose weights, *i.e.*, the FIR taps, are trained individually on a per-class basis.
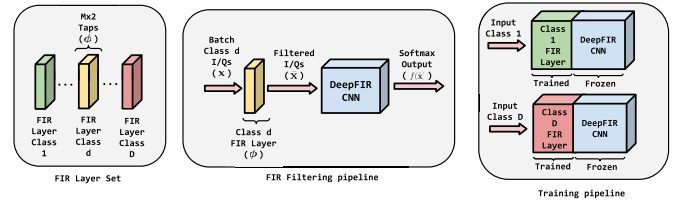

Fig. 6. FIR layer structure, filtering and training pipeline for data-driven FIR computation.

Fig. 7. DeepFIR Experimental Testbed.

Figure 6 shows a simplified block scheme summarizing our data-driven methodology to compute FIR filters. We generate D FIR Layers, each consisting of $M \times 2$ weights (*i.e.*, real and imaginary part of the M FIR taps), *i.e.*, $\phi = (\phi_k^R, \phi_k^I)_{k=0,\dots,M-1}$. We also load DeepFIR CNN pre-trained weights. As shown in Figure 6 (right), only FIR Layer weights ϕ are trained, while DeepFIR CNN model's weights are frozen. We can think of each FIR Layer as additional convolutional layer that is attached to the pre-trained DeepFIR CNN model.

Each FIR layer is fed with I/Q samples belonging to a specific class and it is trained such that filtered I/Q samples generated by the corresponding FIR layer maximize the accuracy of the pre-trained frozen model. We remark that *we do not change the loss function L of the pre-trained DeepFIR CNN model*. Instead, and as shown in Figure 6, we train FIR Layer ϕ, use them to manipulate the input $\mathbf{x}$ and fed the filtered input $\hat{\mathbf{x}}$ (computed as in (2)) to DeepFIR CNN. Each class's FIR Layer weights ϕ are trained by minimizing the cross-entropy loss function L:

$$L = \sum_{d=1}^{D} y_d(\hat{\mathbf{x}}) \log(f_d(\hat{\mathbf{x}})) = \sum_{d=1}^{D} y_d(\mathbf{x}) \log(f_d(\mathbf{x}, \phi)) \quad (14)$$

where $\mathbf{x}$ represents the unfiltered input (*i.e.*, a batch of S slices as discussed in Section IV-C), $\hat{\mathbf{x}}$ is the filtered input sequence, $y_d(\mathbf{x})$ is a binary indicator such that $y_d(\mathbf{x}) = 1$ if the input $\mathbf{x}$ belongs to class d, and f_d is the soft-max activation of the last layer of the pre-trained DeepFIR CNN model.

Our approach, although it does not necessarily guarantee a global optimum, enjoys several useful properties: (i) FIR taps are computed on GPUs via machine learning libraries; and (ii) there is no need to interface the trained model with external numeric solvers. Together, these two features makes it possible not only to speed-up FIR computation, but also to train FIR layers over extremely large datasets (such as those presented in Section VIII.

To avoid excessive distortion of input signals, FIR Layer weights are initialized such that the first weight value ϕ_0^R is set to 1 and the remaining taps are set to 0. This initialization in essence represents an identity vector, which returns unchanged
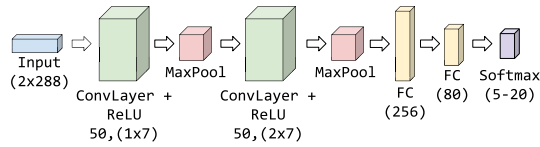
Fig. 8.   CNN used in our radio testbed experiments.

Fig. 9.   PSA and PBA, Experimental Testbed.

input values. To allow for learning and different gradient updates, the 0's are masked with small Gaussian noise. This acts as a form of "personalization" of filters: a device-specific filter is designed to enhance its detectability. This filter is presumed to be attached to the device, and hence is used both in the training and testing phases. This enables the filters to learn channel effects in the scope of individual devices.

VII. RADIO TESTBED RESULTS

A. Wireless Testbed and Data Collection

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 [37] and one VERT2450 antennas [38]. 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.

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 [34]. 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.

B. Deep Learning Architecture

We use the CNN architecture reported in [7] and depicted in Figure 8. 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.

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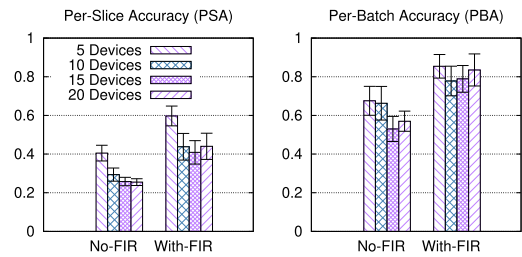
a learning rate of $l = 0.0001$. We minimize the prediction error through back-propagation, using categorical cross-entropy as a loss function computed on the classifier output. To train our CNN, we constructed a dataset by performing 10 different transmissions of length 5 minutes for each of the 20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

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 IV-C, 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 *DeepFIR* is computed on one slice, the PSA measures how much *DeepFIR* 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 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.

C. Results

Figure 9 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 9 show that *DeepFIR* 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 are computed on a testbed of nominally-identically radios, we consider these results remarkable.

The intuition of why PBA is lower than the PSA is that the FIR filter is optimized for a single slice only, thus its
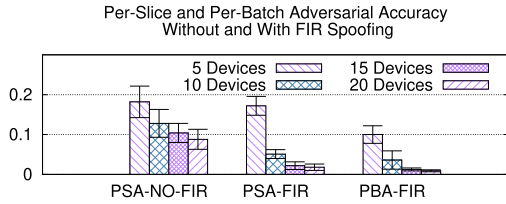
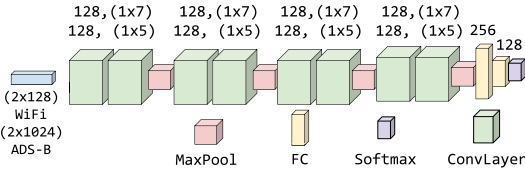Fig. 10.   Adversarial Action, Experimental Testbed.

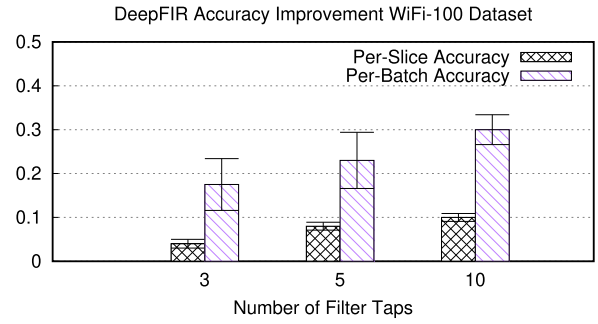Fig. 11.   CNN used in our fingerprinting dataset experiments.

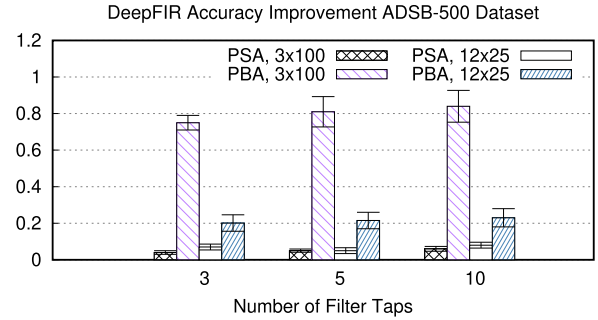Fig. 12.   PSA/PBA Improvement, WiFi-100.

Fig. 13.   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).

efficacy is expected to decrease as time passes and thus the wireless channel changes accordingly. Figure 10 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 10 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.

VIII. Fingerprinting Dataset Results

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, and (iii) the RadioML 2018.01A dataset. The dataset is publicly available for download at http://deepsig.io/datasets.

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 *DeepFIR*'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 VII-C we demodulated the transmissions and trained our models on the derived I/Q samples. To demonstrate the generality of *DeepFIR*, the ADSB model was instead trained on the unprocessed I/Q samples.

To handle the increased number of devices, we use the CNN architecture shown in Figure 11. 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.

To compare our approach with the state of the art by Vo-Huu *et al.* [16], 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 12 shows the PSA and PBA improvement brought by *DeepFIR* 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 [16] which is 47%.

Figure 13 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 13 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 *DeepFIR* to perform better on ADSB-500 when the number of inputs per slice increases.

This is also confirmed by Figure 14, 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 14 shows a similar PBA improvement when 10 FIR taps are considered. Furthermore, Figure 14 shows 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
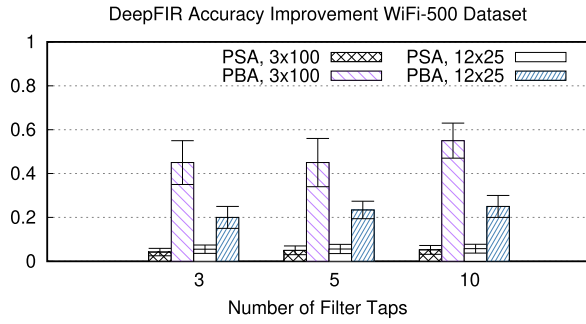
Fig. 14. Per-Slice and Per-Batch Accuracy Improvement, WiFi-500, as a function of the number of inputs per slice (100 and 25) and the number of batches (3 and 12).
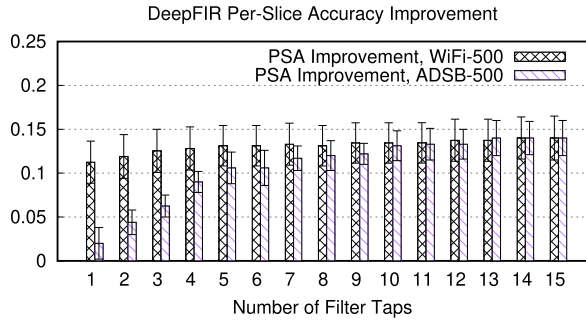

Fig. 15. PSA Improvement as a function of the number of FIR taps. ADSB-500 converges slower than WiFi-500 given the FIR has to operate on unprocessed I/Q samples.

Fig. 16. Average FIR Epsilon (ϵ) values as a function of the number of FIR taps.
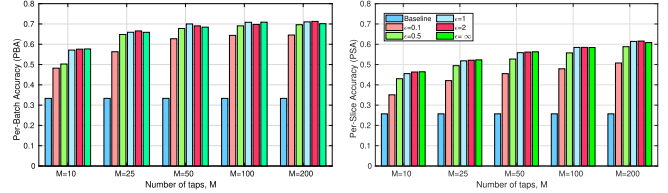
Fig. 17. PSA and PBA as a function of ϵ for different number of FIR taps.

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 15 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 15 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 *DeepFIR* needs to obtain the same performance as in WiFi-500 increases, as the FIR has to compensate for a significant channel action in WiFi-500. Finally, Figure 16 depicts the average FIR ϵ-value (defined in Section V) as a function of the number of taps and the model considered. As anticipated in Section V, Figure 16 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 longer input (1024 vs 128 I/Q samples).

IX. Modulation Dataset Results

In the following, we show the experimental results obtained on the RadioML 2018.01A dataset, which presents 24 different analog and digital modulations with different levels of signal-to-noise ratio (SNR) [3]. For comparison purposes,
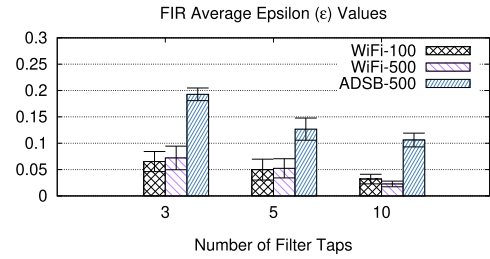
we consider the neural network in [3], Table III. This network is composed by 7 convolutional layers each followed by a MaxPool-2 layer, finally followed by 2 dense layers and 1 softmax layer. The dataset contains 2M examples, each 1024 I/Q samples long.

Figure 17 shows the PBA and PSA with and without Deep-FIR for a variety of different configurations of ϵ and number of FIR taps M. As expected, DeepFIR always improves both PBA and PSA in any configuration of ϵ and M. Similarly to prior results, the accuracy increases as the number of FIR taps increases until it reaches a plateau. We notice that both PSA and PBA increase when large values of ϵ are considered (*e.g.*, the BER constraint is relaxed). In this case, DeepFIR focuses on maximizing the accuracy only disregarding the impact of FIR taps on the BER of the communications. However, even when $\epsilon = 0.1$, the PBA obtained via DeepFIR is doubled when compared to the baseline model.

To understand the impact of DeepFIR on the accuracy of the model, in Figure 18 we show the confusion matrices for different DeepFIR configurations. While the baseline model misclassifies the majority of modulations, DeepFIR considerably improves the accuracy of the network. This is demonstrated by the presence of high values in the main diagonal of the confusion matrix. Although the best results are obtained when both ϵ and M are large, the configuration with $\epsilon = 0.1$ (*i.e.*, DeepFIR computes FIR taps that can be successfully compensated at the receiver side, thus effectively limiting the impact of FIR filtering on the BER of the system), already achieves high accuracy (approximately 65% PBA from Figure 17 when compared to 33% PBA of the baseline model).

To better understand how DeepFIR impacts waveforms and displaces I/Q samples in the complex space, in Figs. 19, 20 and 21 we show waveforms before and after filtering for BPSK, QPSK and 64QAM modulations (i.e., classes 4,5 and 15), respectively. Unless otherwise stated, we consider $\epsilon = 0.1$ and $M = 10$.
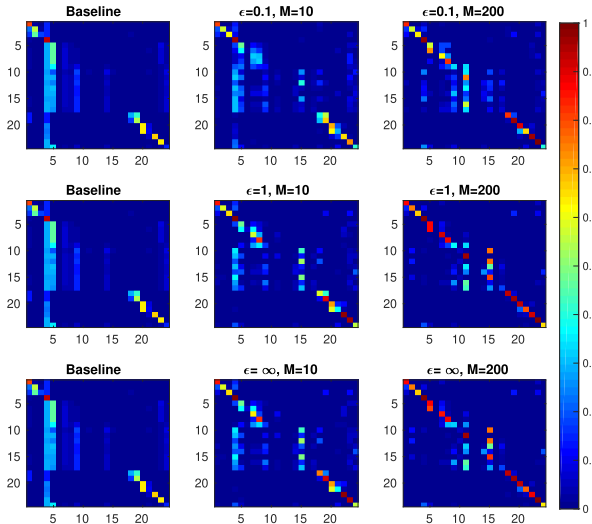
Fig. 18.   Confusion matrices for different ϵ values and number of FIR taps.
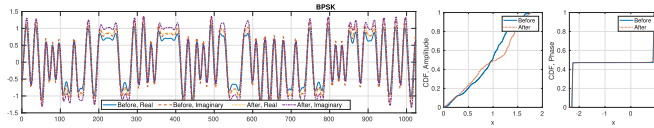

Fig. 19.   Impact of FIR filtering on waveforms and statistical distribution of I/Q samples for BPSK (Class 4) modulation.
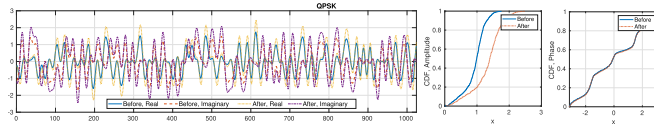

Fig. 20.   Impact of FIR filtering on waveforms and statistical distribution of I/Q samples for QPSK (Class 5) modulation.
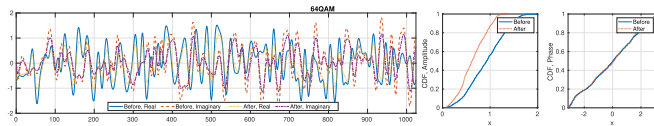

Fig. 21.   Impact of FIR filtering on waveforms and statistical distribution of I/Q samples for 64QAM (Class 15) modulation.

Figure 19 shows that FIR taps computed by DeepFIR are leaving the phase of the signal unchanged and are only minimally varying their amplitude. This is an interesting result as BPSK signals are demodulated by determining the phase of received I/Q samples. This way, a receiver would still be able to demodulate received BPSK waveforms properly without errors. The same also holds for QPSK modulated signals. As an example Figure 20 shows that DeepFIR only affects the amplitude of transmitted I/Q samples while leaving the phase untouched (note the characteristic step CDF of the phase).

Figure 21, instead, shows the waveform for a 64QAM signal before and after DeepFIR filtering with $\epsilon = 1$ and $M = 10$. In this particular configuration, the two CDFs show that the impact of DeepFIR on the phase is slightly more relevant than in the case of PSK modulations, while the amplitude of the signal is reduce for most I/Q samples. Despite this reduction might seem destructive, recall that we can always compensate
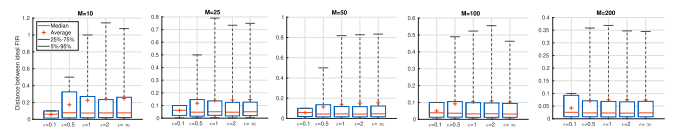

Fig. 22.   Statistical analysis of data-driven FIR taps.

waveform alterations, especially for the considered setup with $\epsilon = 1$ and $M = 10$, meaning that the impact of DeepFIR on the demodulation procedure is negligible.

Finally, in Figure 22 we present results on the statistical distribution of computed FIR taps. Recall that ϵ represents the maximum distance between FIR taps and the ideal FIR (i.e., $\phi = (1, 0, \ldots, 0)$). Figure 22 shows that the data-driven approach (i) always satisfies the maximum distance constraint; and (ii) on average, the majority of taps shows a maximum distance from the ideal FIR less than 0.2. This is an important result because, as we have discussed in Section V, the receiver can effectively compensate FIR action and recover the original transmitted waveform.

X. CONCLUSION

In this paper, we have proposed *DeepFIR*, a methodology to make deep-learning-based waveform classification algorithms channel-resilient. *DeepFIR* leverages a finite input response filter (FIR) at the transmitter's side to compensate for current channel conditions, which is computed by the receiver through optimization and sent back as feedback to the transmitter. We have extensively evaluated *DeepFIR* on a experimental testbed of 20 nominally-identical software-defined radios, as well as on datasets made up by WiFi and ADS-B transmissions and a 24-class modulation dataset. Experimental results have shown that *DeepFIR* is significantly effective in increasing the classification accuracy of existing state-of-the-art deep learning models.

ACKNOWLEDGMENT

REFERENCES

[1] J. Jagannath, N. Polosky, A. Jagannath, F. Restuccia, and T. Melodia, "Machine learning for wireless communications in the Internet of Things: A comprehensive survey," *Ad Hoc Netw.*, vol. 93, Oct. 2019, Art. no. 101913.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[3] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 168–179, Feb. 2018.

[4] F. Restuccia and T. Melodia, "Big data Goes small: Real-time spectrum-driven embedded wireless networking through deep learning in the RF loop," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 2152–2160.

[5] T. J. O'Shea, S. Hitefield, and J. Corgan, "End-to-end radio traffic sequence recognition with recurrent neural networks," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Dec. 2016, pp. 277–281.

[6] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury, "ORACLE: Optimized radio classification through convolutional neural networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 370–378.

[7] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep learning convolutional neural networks for radio identification," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 146–152, Sep. 2018.

[8] H. Zhang, W. Li, S. Gao, X. Wang, and B. Ye, "ReLeS: A neural adaptive multipath scheduler based on deep reinforcement learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1648–1656.

[9] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart., 2019.

[10] F. Restuccia *et al.*, "DeepRadioID: Real-time channel-resilient optimization of deep learning-based radio fingerprinting algorithms," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jul. 2019, pp. 51–60.

[11] A. Al-Shawabka *et al.*, "Exposing the fingerprint: Dissecting the impact of the wireless channel on radio fingerprinting," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 646–655.

[12] A. Goldsmith, *Wireless Communications*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[13] M. Sugiyama and M. Kawanabe, *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. Cambridge, MA, USA: MIT Press, 2012.

[14] S. Chetlur *et al.*, "CuDNN: Efficient primitives for deep learning," 2014, *arXiv:1410.0759*. [Online]. Available: http://arxiv.org/abs/1410.0759

[15] K. Sankhe *et al.*, "No radio left behind: Radio fingerprinting through deep learning of physical-layer hardware impairments," *IEEE Trans. Cognit. Commun. Netw.*, vol. 6, no. 1, pp. 165–178, Mar. 2020.

[16] T. D. Vo-Huu, T. D. Vo-Huu, and G. Noubir, "Fingerprinting Wi-Fi devices using software defined radios," in *Proc. 9th ACM Conf. Secur. Privacy Wireless Mobile Netw.*, Jul. 2016, pp. 3–14.

[17] F. Restuccia and T. Melodia, "Physical-layer deep learning: Challenges and applications to 5G and beyond," 2020, *arXiv:2004.10113*. [Online]. Available: https://arxiv.org/abs/2004.10113

[18] N. T. Nguyen, G. Zheng, Z. Han, and R. Zheng, "Device fingerprinting to enhance wireless security using nonparametric Bayesian method," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2011, pp. 1404–1412.

[19] L. Peng, A. Hu, J. Zhang, Y. Jiang, J. Yu, and Y. Yan, "Design of a hybrid RF fingerprint extraction and device classification scheme," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 349–360, Feb. 2019.

[20] M. Bkassiny, Y. Li, and S. K. Jayaweera, "A survey on machine-learning techniques in cognitive radios," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1136–1159, 3rd Quart., 2013.

[21] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 98–105, Apr. 2017.

[22] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks," *CoRR*, vol. abs/1710.02913, pp. 1–84, Oct. 2017.

[23] M. L. D. Wong and A. K. Nandi, "Automatic digital modulation recognition using spectral and statistical features with multi-layer perceptrons," in *Proc. 6th Int. Symp. Signal Process. Appl.*, vol. 2, 2001, pp. 390–393.

[24] J. L. Xu, W. Su, and M. Zhou, "Software-defined radio equipped with rapid modulation recognition," *IEEE Trans. Veh. Technol.*, vol. 59, no. 4, pp. 1659–1667, May 2010.

[25] S. U. Pawar and J. F. Doherty, "Modulation recognition in continuous phase modulation using approximate entropy," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 843–852, Sep. 2011.

[26] Q. Shi and Y. Karasawa, "Automatic modulation identification based on the probability density function of signal phase," *IEEE Trans. Commun.*, vol. 60, no. 4, pp. 1033–1044, Apr. 2012.

[27] S. Ghodeswar and P. G. Poonacha, "An SNR estimation based adaptive hierarchical modulation classification method to recognize M-ary QAM and M-ary PSK signals," in *Proc. 3rd Int. Conf. Signal Process., Commun. Netw. (ICSCN)*, Chennai, India, Mar. 2015, pp. 1–6.

[28] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2595–2621, Jun. 2018.

[29] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cognit. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017.

[30] T. Wang, C.-K. Wen, H. Wang, F. Gao, T. Jiang, and S. Jin, "Deep learning for wireless physical layer: Opportunities and challenges," *China Commun.*, vol. 14, no. 11, pp. 92–111, Nov. 2017.

[31] N. E. West and T. O'Shea, "Deep architectures for modulation recognition," in *Proc. IEEE Int. Symp. Dyn. Spectr. Access Netw. (DySPAN)*, Baltimore, MD, USA, Mar. 2017, pp. 1–6.

[32] M. Kulin, T. Kazaz, I. Moerman, and E. De Poorter, "End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications," *IEEE Access*, vol. 6, pp. 18484–18501, 2018.

[33] K. Karra, S. Kuzdeba, and J. Petersen, "Modulation recognition using hierarchical deep neural networks," in *Proc. IEEE Int. Symp. Dyn. Spectr. Access Netw. (DySPAN)*, Baltimore, MD, USA, Mar. 2017, pp. 1–3.

[34] E. Sourour, H. El-Ghoroury, and D. McNeill, "Frequency offset estimation and correction in the IEEE 802.11a WLAN," in *Proc. IEEE 60th Veh. Technol. Conf. (VTC-Fall)*, vol. 7, Sep. 2004, pp. 4923–4927.

[35] E. Johnson, "Physical limitations on frequency and power parameters of transistors," in *Proc. IRE Int. Conv. Rec.*, vol. 13, 1966, pp. 27–34.

[36] M. R. Hestenes and E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, vol. 49. Washington, DC, USA: NBS, 1952.

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

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

Francesco Restuccia (Member, IEEE) is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Northeastern University, USA. He has published over 30 articles in top venues, and coauthoring more than ten pending U.S. patents and two book chapters. His research interests include the modeling, analysis, and experimental evaluation of wireless networked systems. He is a member of the ACM. He regularly serves as a TPC Member for conferences, such as IEEE INFOCOM and ACM MobiHoc. He was a recipient of the 2019 ISSNAF Mario Gerla Award for Young Investigators in Computer Science. He is a reviewer for several ACM and IEEE conferences and journals.
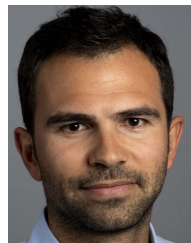
Salvatore D'Oro (Member, IEEE) received the Ph.D. degree from the University of Catania in 2015. He is currently an Associate Research Scientist with Northeastern University. He served on the Technical Program Committee (TPC) for the IEEE Conference on Standards for Communications and Networking (CSCN'18), Med-Hoc-Net 2018, and the CoCoNet8 Workshop at IEEE ICC 2016. In 2015, 2016, and 2017, he organized the 1st, 2nd, and 3rd Workshops on COmpetitive and COoperative Approaches for 5G Networks (COCOA). He is a reviewer for major IEEE and ACM venues.

Amani Al-Shawabka received the bachelor's degree in communication engineering from Yarmouk University, Jordan, and the master's degree in computer engineering from Northeastern University in 2019, where she is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. She started her professional career at mobile network companies after completing the bachelor's degree. Her research interests include wireless networks, embedded systems, and network security.

Bruno Costa Rendon received the master's degree in machine learning from the College of Electrical and Computer Engineering, Northeastern University. He is currently working with Raytheon on machine learning projects. His research interests include machine learning and computer vision.

Tommaso Melodia (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology in 2007. He is currently a William Lincoln Smith Chair Professor with the Department of Electrical and Computer Engineering, Northeastern University, Boston. He is also the Founding Director of the Institute for the Wireless Internet of Things and the Director of Research for the PAWR Project Office. He is a Senior Member of the ACM. He was a recipient of the National Science Foundation CAREER Award. He has served as the Technical Program Committee Chair for IEEE Infocom 2018, the General Chair for IEEE SECON 2019, ACM Nanocom 2019, and ACM WUWnet 2014. His research on modeling, optimization, and experimental evaluation of the Internet of Things and wireless networked systems has been funded by the National Science Foundation, the Air Force Research Laboratory, the Office of Naval Research, DARPA, and the Army Research Laboratory. He has served as an Associate Editor for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON MOBILE COMPUTING, and *Computer Networks* (Elsevier).

Stratis Ioannidis received the B.Sc. degree in electrical and computer engineering from the National Technical University of Athens, Greece, in 2002, and the M.Sc. and Ph.D. degrees in computer science from the University of Toronto, Canada, in 2004 and 2009, respectively. He was a Research Scientist with the Technicolor Research Centers in Paris, France, and Palo Alto, CA, USA, and Yahoo Labs, Sunnyvale, CA, USA. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Northeastern University. His research interests include machine learning, distributed systems, networking, optimization, and privacy. He was a recipient of the NSF CAREER Award, the Google Faculty Research Award, and the Best Paper Award at ACM ICN 2017.