More is Better: Data Augmentation for Channel-Resilient RF Fingerprinting

Nasim Soltani, Kunal Sankhe, Jennifer Dy, Stratis Ioannidis, and Kaushik Chowdhury
Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, USA
Emails: {soltani.n,sankhe.ku}@northeastern.edu,{jdy,ioannidis,krc}@ece.neu.edu

***Abstract*—RF fingerprinting involves identifying characteristic transmitter-imposed variations within a wireless signal. Deep Neural Networks (DNNs) that do not rely on handcrafting features have proven to be remarkably effective in fingerprinting tasks, as long as the channel remains invariant. However, DNNs trained at a specific location and time perform poorly on datasets collected under different channel conditions. This paper proposes a data augmentation step within the training pipeline that exposes the DNN to many simulated channel and noise variations that are not present in the original dataset. We describe two approaches for data augmentation: The first approach is applied to the 'transmitter data' when transmitter side data (i.e., pure signals without channel distortion) is available. The second approach is applied to the 'receiver data', when only a passive dataset is available with already over-the-air transmitted signals. We show that data augmentation results in 75% improvement in the former case with a custom-generated dataset, and around 32-51% improvement in the latter case on a 5000-device WiFi dataset, compared to the case of non-augmented data fed to DNNs.**

I. Introduction

RF fingerprinting is a process to identify radios by detecting a characteristic signature embedded within their transmitted electromagnetic waves. Traditionally, this process involves handcrafting features that represent salient hardware characteristics manifesting in the transmission. However, identifying the most discerning features from a pool comprising multitude of physical radios is challenging. It is protocol specific and thus requires domain knowledge and advanced test equipment. As opposed to this, deep-learning-based methods are gaining traction due to their ability to automatically identify hardware features in a protocol-independent fashion: only raw in-phase (I) and quadrature (Q) components of the samples suffice for detection, which considerably simplifies the end-user application of RF fingerprinting. While many works [1]–[3] have made substantial strides in radio fingerprinting using raw IQ samples with DNNs, they report classification accuracy only with datasets where the training set and the test set are collected under very similar wireless channels and environmental conditions.

Wireless channel is a major contributor to accuracy degradation in DNN-based RF fingerprinting. It effectively scales up/down and rotates the IQ constellation due to attenuation, reflections, and delays. These highly complex interactions are unique to a particular channel and may not repeat in exactly the same way in future channel conditions. Therefore, with a training set collected under particular channel conditions, the DNN ultimately ends up learning a channel-distorted fingerprint instead of the pure inherent fingerprint. A DNN trained thusly yields poor accuracy if tested under a different channel. As evidence, our prior work ORACLE [2] shows 99% classification accuracy for 16 software defined radios (SDRs) when training and test sets are collected under the same channel conditions. However, this accuracy drops to 56% when the DNN is tested under a different channel.

Popular methods to overcome the channel effect are transfer learning [4], [5] and re-training [4]. These solutions are not always possible, as training during deployment is resource and time consuming. Therefore, a means of making the neural network resilient to unseen channel and noise variations is of paramount importance. More about these methods are discussed in the next Section.

In image processing domain, a common approach to make the neural network resilient to a specific type of variation and avoid overfitting is *data augmentation* [4], that is, expanding the training set with additional samples that resemble the outcomes of that variation. For example, geometric transformations such as rotating, flipping and re-scaling the training samples are common in image processing [4]; so is adding salt-and-pepper and Gaussian noise [6]. However, these image transformations do not account for the inherent properties of wireless signals, and are not suitable for wireless domain.

The main contribution in this paper is to propose a novel methodology for data augmentation in the RF domain. In our method, the training data is augmented in a principled manner that makes the trained DNN resilient to channel variations and noise levels. Data is sequentially passed through a *channel model* and a *noise model*. The *channel model* is an FIR filter –with filter taps drawn from a specific distribution– being convolved with the signal passing through it. The *noise model* is a noise generator, producing random values from a Gaussian distribution with variance proportional to the noise power. These random values are summed with the output of the *channel model*. The DNN trained with the augmented training set yields a channel-and-noise-resilient neural network for RF fingerprinting.

Our contributions are as follows:

- We propose a data augmentation method integrated within a deep learning pipeline for channel-resilient RF fingerprinting on both cases of 'transmitter data' (i.e., transmitter data accessible) and 'receiver data' (i.e., only a passive received dataset is available). The DNN trained with this approach performs accurate RF fingerprinting even under unseen channels.

- We provide a simulated dataset generated in MATLAB using WiFi 802.11a PHY frames for 10 virtual transmitters and different SNR levels. In addition, we show how 'receiver-side' augmentation improves classification accuracy in a DARPA-provided 5000 WiFi device dataset.
- For receiver-side augmentation, we provide a discussion on the strategies for selecting the augmentation parameters that retain the scale of the original signals, so that the normalized test set can be fed to the DNN without any augmentation. We also highlight the open research challenges in this area.

II. Related Work

Among the large body of works exploiting DNNs for RF fingerprinting, we survey those approaches that attempt to overcome the drop in classification accuracy when the channel changes between training and test sets.

ORACLE [2] works on the 'receiver-side' data, where it equalizes the received data before forming training and test sets. Equalization estimates and compensates for the effect of the channel. However, this approach needs full knowledge of the waveform (i.e., modulation, sampling rate and frame structure). Furthermore, it requires preprocessing the IQ samples, which increases delays. The data augmentation method proposed in this paper, instead, can be applied to raw IQ samples without any prior knowledge about the waveform.

DeepRadioID [7] finds an FIR filter at the 'transmitter-side' to negate the channel. The FIR filter at the transmitter-side is optimized based on the current channel conditions and the transmitter's characteristics to synthesize a filtered waveform. The overall outcome of this step is that the FIR filter makes the transmitters more distinguishable to the trained Convolutional Neural Network (CNN). However, a new FIR filter must be computed for each transmitter every time the channel changes. This step is computationally heavy as it relies on back-propagation within the trained CNN to identify the optimal filter taps. Moreover, it needs a reliable backchannel to communicate filter taps obtained at the receiver to the transmitter. As opposed to this, data augmentation neither requires any live processing in the field, nor any receiver-transmitter coordination.

Data augmentation specifically for wireless using Generative Adverserial Networks (GANs) is proposed in [8]. The authors introduce variations in the training set by generating synthetic data that resemble the original training set. However, GANs may not be suitable to train channel-resilient neural networks, since the channel-distorted signals do not resemble the original data. Instead, our data augmentation scheme creates distortions similar to channel and noise-induced effects in the original training set.

III. Dataset Generation and Training the DNN

In this section, we describe the two datasets used in this paper, the steps for data generation and pre-processing, and the neural network architectures.

A. Custom-Generated Dataset

We use MATLAB Communications Toolbox to simulate 10 *virtual* radios. We use a classical transmitter chain and modify it by introducing RF impairments that are seen in actual radio hardware. We set different levels of IQ imbalance, and each choice results in one distinct virtual radio (simply abbreviated as r_1 to r_{10}). While real radios have a combination of impairments, we focus on IQ imbalance as described in [2]. RF fingerprinting aims to distinguish these 10 radios using the received IQ samples. To create 10 virtual radios, we vary the amplitude imbalance from 1 to 5.5dB with steps of 0.5dB and phase imbalance from 1 to 82 with steps of 9 degrees. Average bit error rate for these IQ imbalance values is 0.0031 for SNR>4dB, which ensures the impairments do not disrupt the communication [2], [3].

Each radio transmits IEEE 802.11a WiFi frames generated via MATLAB WLAN toolbox. For each payload, we modulate a random bit sequence with QPSK modulation scheme and $1/2$ coding rate. These packets, unmodified by the wireless channel, are recorded at the transmitter-side. We refer to this dataset as *TxData* from here on.

Next, we simulate different instances of an indoor wireless channel using a 9-tap `WlanTgn` channel model implemented in WLAN toolbox. Different instances of `WlanTgn` channel are obtained by varying the 'channel seed' for each transmission. We vary the SNR from -10 to 20dB with steps of 2dB by changing the Additive White Gaussian Noise (AWGN) level. At each SNR level, a given radio transmits WiFi packets over a specific instance of the channel until we collect 19.6 million IQ samples from that radio at the receiver-side. This process is performed for 10 radios at 16 SNR levels. We refer to this dataset as *Day1*, emulating the captured transmissions from 10 radios on a given day. We further repeat this one more time, providing dataset *Day2*. This entire custom dataset including *TxData*, *Day1* and *Day2* is available in our collection [9].

B. DARPA Dataset

Our simulated dataset described above is used to demonstrate data augmentation at the transmitter-side. However, in many situations, we have access only to raw IQ samples at the receiver-side, which have already traversed a wireless channel. To show how augmentation works in this case, we use a dataset provided by DARPA. While this dataset has dissemination restrictions, it contains signals from 50, 250, 500 and 5000 WiFi devices transmitting IEEE 802.11a/g protocol. These datasets are collected 'in the wild' and contain on average 166 examples per device. There are 10900 to 110000 examples in the training set, and 2750 to 30000 examples in the test set, depending on the number of devices. Each example corresponds to an independent transmission and has an average length of $\sim$18k IQ samples.

C. DNN Architectures and Training

We use 3 different DNN architectures in this paper, which are shown in Figure 1. Among them, CNN1 is a feed-forward Convolutional Neural Network with $\sim$1.1M parameters that

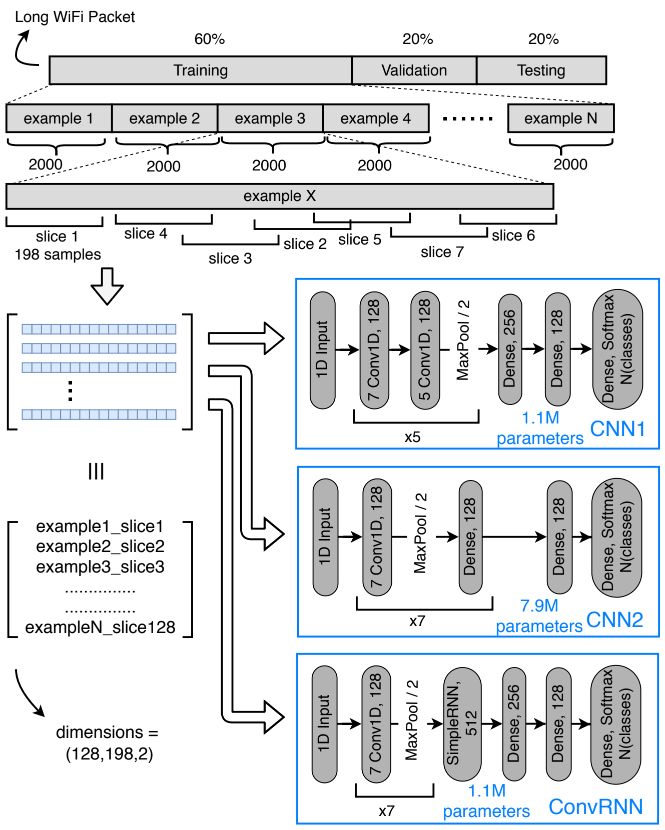

Figure 1. Forming tensors for the neural networks and 3 different neural network architectures of CNN1, CNN2 and ConvRNN.

previously performed well for RF fingerprinting [10] and modulation classification [11]. CNN2 is a more complex feed-forward Convolutional Neural Network with ∼7.9M parameters and ConvRNN is a Convolutional Recurrent Neural Network with a `SimpleRNN` layer and ∼1.1M parameters. We train the neural networks using Adam optimizer with a learning rate of 0.0001.

1) Learning on the Custom Dataset: We next describe the data preprocessing steps before feeding the IQ samples to the DNN. The dataset corresponding to any given *Day* consists of signal transmissions from 10 radios collected at a fixed SNR from a total of 16 distinct levels. Each of these transmissions comprises a sequence of 19.6 million IQ samples. Thus, we have $10 \times 16 = 160$ sequences for each *Day*. We partition each sequence into non-overlapping sets of training (60%), validation (20%) and test (20%). Each set is further divided into several non-overlapping examples of length L to form independent transmissions. Each example yields $L - l + 1$ overlapping subsequences, referred to as *slices*, by sliding a window of length l along it [10]. The sliding window approach enhances the shift invariance of the features learnt by the DNN [10]. We set each example to be of size $L = 2000$ to ensure it is long enough to yield multiple *slices* of length $l = 198$.

During training, we load a set of examples using `Data Generator` class from `Keras` library. Inside the `Data Generator`, 128 random slices with length 198 are chosen from random examples, to form a data batch. The random selection of examples and slices in every epoch contributes to training more robust deep learning models [10]. Each data batch forms a tensor with dimension $(128, 198, 2)$, where I and Q information is included via separate channels, in the last dimension (see Figure 1).

We train CNN1 in Figure 1 with the training set from *Day1* and test it on the test set from *Day1* and then *Day2*. We calculate per slice accuracy by dividing the number of correctly predicted slices by the total number of slices. We classify each example by summing the probability vectors of all the slices in that example and choosing the class with the highest value as the predicted class. We calculate per example accuracy by dividing the number of correctly predicted examples by the total number of test examples.

2) Learning on the DARPA Dataset: Since in the DARPA dataset the non-overlapping examples are already formed, tensors are extracted out of the examples in the same manner as our custom dataset. More details are discussed in previous work [10].

IV. Data Augmentation

The purpose of data augmentation in the training pipeline is to make the DNN robust to channel and noise variations in the test set. In this process, the training data is passed through an augmentation block that captures different virtual instances of the wireless channel and the receiver noise. Data augmentation can be performed either on the transmitter data, or the received IQ samples. After the network is trained, classifying the radios (i.e., the test phase) happens using a test set containing received IQ samples.

A. Data Augmentation on the Transmitter Data

For data augmentation on the transmitter data, no changes need to happen in the transmitter processing chain. Instead, transmitter data sequences –that contain the transmitter fingerprint, but no channel or noise distortions– are recorded to train the neural network. Sequences are chopped into non-overlapping examples and data batches are created out of examples (as described in the previous section) using `Keras Data Generator`. In the classical approach for training, we simply feed these batches to the DNN. However, with our data augmentation scheme, each batch passes through an augmentation block before being fed to the DNN (see Figure 2). The augmentation block comprises: (i) a channel model and (ii) a noise model.

1) ***Channel Model****:* The channel model is a mathematical representation of a wireless channel that essentially captures the effects of natural distortions (e.g., multipath fading on a transmitted wireless signal). The multipath fading channel is often modeled with a multi-tap FIR filter with appropriate channel frequency response [12].

We use Wireless LAN TGn (`WlanTgn`) channel model with delay profile of type Model-B. This model characterizes a typical indoor, large open space and office environment that has non line-of-sight wireless propagation of 15ns rms delay spread [13]. The model is simulated using 9 taps of complex channel coefficients, representing path gains and

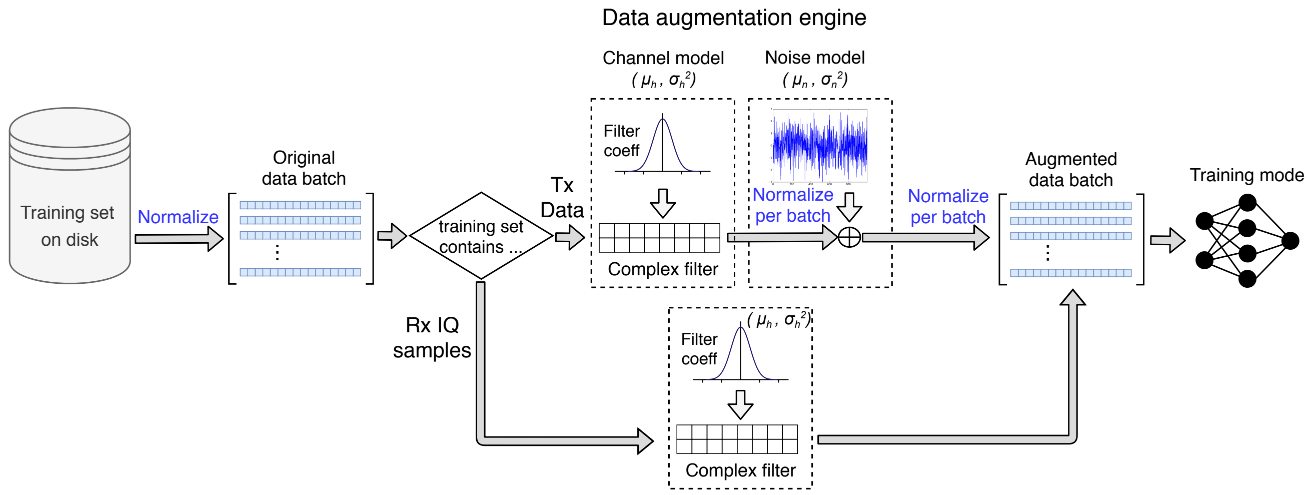

Figure 2. Data augmentation is performed in the training phase. The pipeline for augmentation on the TxData comprises the channel model and the noise model. The pipeline for augmentation on the Rx IQ samples comprises the complex FIR filter.

path delays. Following the central limit theorem, we sample these coefficients from a Complex Gaussian distribution with mean μ_h and variance σ_h^2. Parameters μ_h and σ_h^2 can be estimated from different realizations of `WlanTgn` channel model. However, in data augmentation on the transmitter data, μ_h and σ_h^2 are compensated by normalizing the data batch, at the input of the noise model. Therefore, we use typical values of $\mu_h = 0$ and $\sigma_h^2 = \frac{1}{2\times 9}$, to ensure total power distribution of 9 complex taps equals 1 unit.

During training, in every epoch, per batch of size $(128, 198, 2)$, a new set of 9 complex taps (representing the channel model) are independently drawn from Gaussian distribution. Each slice is convolved with this FIR filter. The output slices are stacked together to form a batch with the same dimension as the input batch $(128, 198, 2)$. The output batch is thus the transmitter IQ samples passed through wireless channel model. Choosing new filter taps per batch and per epoch exposes the DNN to hundreds of thousands of different channel instances during training. While with classical training, validation accuracy saturates after several epochs, with data augmentation the accuracy keeps improving as we continue training.

*2) **Noise Model**:* After the data batch passes through the channel model, it is fed to the noise model that emulates the additive receiver noise. The level of noise is chosen based on the SNR variations we expect in the test set. In our case, our objective is to study how robust the DNN is to SNRs in range $[-10, 20]$dB with steps of 2dB.

In the noise model, first the data batch is normalized to ensure power $= 1$. Next, an SNR value is randomly drawn from the above range, which determines the power (variance σ_n^2) of noise. Then, a batch of noise with the same dimensions as the input is generated from Gaussian distribution with mean $\mu_n = 0$ and variance σ_n^2 inversely proportionate to the SNR level.

The batch of white Gaussian noise is finally summed with the filtered batch of signal. This completes the process of distorting the signal by both channel and noise models. We ensure that the resulting batch of data is always normalized before being fed to the DNN. It should be noted that in data augmentation on the transmitter data, the training set is never passed through a simulated channel in MATLAB. Instead, the channel and noise are modeled in data augmentation engine in the deep learning framework, as explained earlier.

In the test phase, since the received IQ samples already passed through simulated channel and noise in MATLAB, no further processing is needed in the deep-learning pipeline. Test data batches only need to be normalized before entering the DNN, as the DNN is trained with normalized data batches.

B. Data Augmentation on the Receiver Data

The complex FIR filter in the data augmentation block can be used also for augmentation on the receiver data. In this case, the convolution of the FIR filter does not conceptually reflect the action of the wireless channel. The main contribution of the filter, instead, is to provide substantial variety in the training set by distorting the received IQ samples through a random selection of FIR taps per data batch. This variety in the training set, prevents the DNN from overfitting and, hence, improves the test accuracy.

In data augmentation on the receiver data, similar to a conventional classification problem, we normalize the training set across all IQ samples in it. Then we load training batches and pass them through a randomly chosen FIR filter with 11 complex taps (see Figure 2). Similar to “Data Augmentation on Tx Data”, the FIR taps are drawn from Complex Gaussian distribution with mean μ_h and variance σ_h^2. Here, to prevent the scale of training batches from changing after filtering, we consider μ_h and σ_h^2 for an 11-tap complex *Identity* filter. This filter has one element with real part ‘1’ and imaginary part ‘0’, and ten elements equal to ‘0’. For this filter, $\mu_h = 0.045$ and $\sigma_h^2 = 0.0434$. We use these statistics for the Gaussian FIR filter in the training pipeline, so that the scale of training data does not change after filtering. In this way, the normalized test set can be fed to the DNN without passing through a filter.

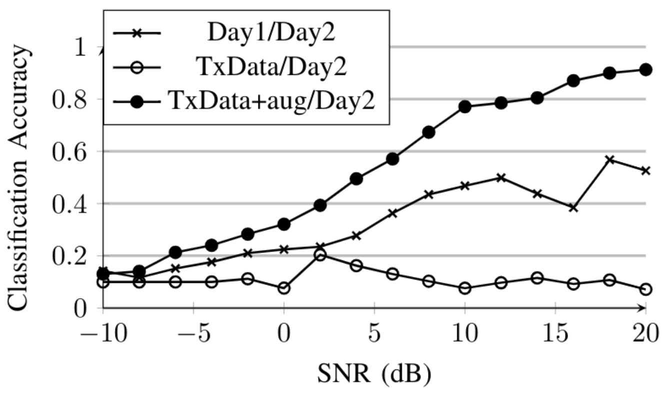

Figure 3. Classification accuracy versus SNR for different cases of TrainingSet/TestSet.

V. Evaluation

In this section, we show the accuracy drop when the test set is collected under unseen channels, and how data augmentation presents a viable solution. For data augmentation on the transmitter data, we use the custom-generated dataset with CNN1 in Figure 1. For data augmentation on the receiver IQ samples, we use the DARPA dataset with CNN1, CNN2 and ConvRNN shown in Figure 1.

A. Accuracy Drop with Unseen Channels

Here, we quantitatively demonstrate the drop in classification accuracy if we train the CNN1 on one *Day* but test it with data from another *Day*. When we train the DNN with the training set from *Day1* (described earlier in Custom-Generated Dataset subsection) and test it with test set from *Day1*, classification accuracy is 99% for SNRs>12dB. Thus, the virtual radios can be well distinguished when the training and test sets are collected using the same wireless channel. Next, after training CNN1 with *Day1*, we test it using *Day2*. Classification accuracy versus SNR for this case is shown in Figure 3 as 'Day1/Day2'. We see that the accuracy drops to 52% even in the comparatively high SNR=20dB. This is because when we train the model with *Day1*, we are in part learning the wireless channel along with the radio fingerprints, which impacts the classification accuracy in generalized, different-day test scenarios.

B. Data Augmentation on Tx Data

To show how data augmentation addresses this problem, we train CNN1 with pure transmitter-side IQ samples before passing through the channel (called as TxData) for 10 radios without data augmentation in the pipeline. The network trained thusly is not able to classify radios from unseen channels (see plot 'TxData/Day2' in Figure 3). We train CNN1 on TxData, this time *with* data augmentation scheme and test it with *Day2*. In this case, the network is able to detect devices from unseen channels and noise levels (see plot 'TxData+aug/Day2' in Figure 3). The resulting 91% accuracy at SNR=20dB shows 75% improvement, compared to 52% for the earlier case of Day1/Day2, when CNN1 is trained on one *Day* and tested on another.

Figure 4 shows the confusion matrices for CNN1 trained with TxData (without data augmentation engine) and with TxData passing through the cascade of the channel model and the noise model. Both trained models are tested with *Day1* data at SNR 20dB. As we can see, if the network is trained with pure transmitter data without the augmentation block and tested with the test set in *Day1*, the classification would be randomly performed. This happens due to the absence of channel and noise variations in the training set. In this case, the confusion matrix does not show any particular pattern. However, if the network is trained with transmitter data with the channel model and the noise model in the pipeline, the highlights around the diagonal of the confusion matrix shape vividly. The diagonal highlights represent each true label being predicted correctly which yields high classification accuracy.

C. Data Augmentation on Rx Data

As described earlier, we use WiFi raw IQ samples from the US Defense Advanced Research Projects Agency (DARPA)-provided dataset, for 50, 250, 500 and 5000 devices. Figure 5 shows per example accuracy without and with data augmentation in the training pipeline, for different dataset sizes. Augmentation in all dataset sizes is validated using CNN1. For the 50-device dataset, two additional DNNs of CNN2 and ConvRNN are also used to show the performance across different architectures. The results for the 50-device dataset show that data augmentation improves accuracy for different DNNs up to 35%.

The overall results for different dataset sizes in Figure 5 show a boost of 35%, 51%, 32% and 41% for 50, 250, 500 and 5000 device datasets, respectively. In these cases, data augmentation prevents overfitting by providing variety in the training set, which boosts the test accuracy.

VI. Open Research Challenges

We identify the following research challenges for the application of data augmentation in the training pipeline in RF fingerprinting:

1. ***Type of filter***: Our data augmentation scheme uses FIR filters that present several advantages: First, they do not rely on future inputs, only past and present ones. Second, they are easy to implement and can approximate a function through appropriate weighting and a finite-term sum. Whether alternate filters such as Infinite Impulse Response (IIR) filters that combine FIR filters with recursive loops also work, is an open question.
2. ***FIR coefficient range***: We showed that data augmentation works without the need to filter the test set if the FIR taps are chosen from Complex Gaussian distribution with specific μ_h and σ_h^2. However, if the test set also passes through an FIR filter with the same statistics as the FIR filter in the training phase, this μ_h and σ_h^2 can vary. Nevertheless, there are permissible upper and lower bounds for choosing the FIR coefficients for each dataset. Going beyond these thresholds may actually reduce the

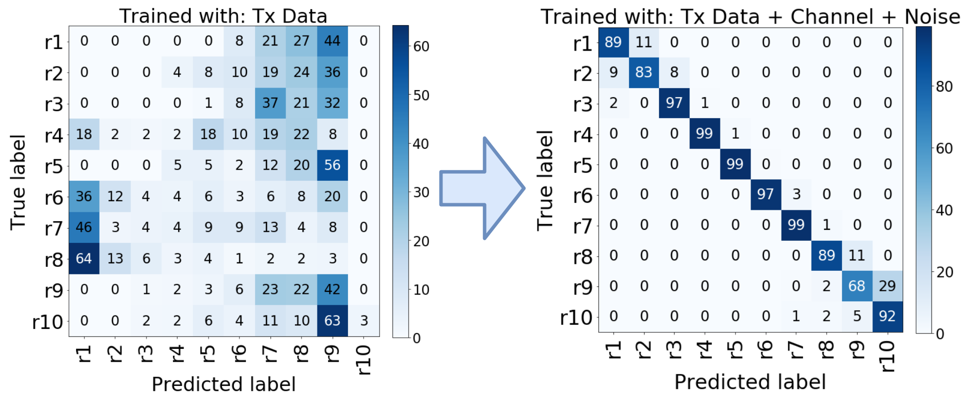

Figure 4. Confusion matrices for augmentation on TxData, trained without and with augmentation engine. Both models are tested with *Day1* data at SNR=20dB.

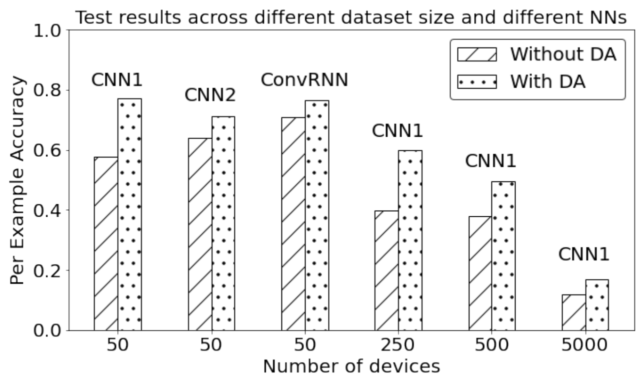

Figure 5. Per example accuracies without and with Data Augmentation (DA) for the DARPA dataset.

accuracy. For example, signals from different classes may be confused with each other after filtering with a set of coefficients with arbitrary variance, which decreases the classification accuracy.

3. ***Number of FIR filter taps***: For data augmentation on the receiver IQ samples, we are not confined to a particular channel model. Hence, the number of taps for the FIR filter can vary to arbitrarily large numbers. We have not yet explored the effect of this parameter on the accuracy.
4. ***Training indoors, testing outdoors***: With our simulated data, we showed data augmentation works, but within the boundaries of a single channel model, even when specific instances of the channel are different. This is analogous to the situation when the same indoor environment is instantiated on different days. However, one of the main challenges in deep-learning-based wireless signal classification is transitioning between environments. It remains an open question if the training dataset collected in static indoor environment, even with extensive data augmentation, can help if testing is done in an outdoor environment.

VII. Conclusion

This paper describes how data augmentation can improve classification accuracy in situations when a DNN is trained with data from one wireless channel and tested on data from another channel. Our data augmentation block works on both pure transmitter-side IQ samples (before transmission over the wireless channel) as well as receiver-side IQ samples (that have gone through a wireless channel). Data augmentation enhances the training set by introducing different distortions resembling instances of the channel and noise. The DNN trained with augmented data is robust to unseen channels and noise variations in the test set. We demonstrated upto 75% and 51% increase in signal classification accuracy over the non-augmented case in a custom dataset and the DARPA dataset, respectively. Thus, we believe data augmentation can help to train channel-resilient DNNs. This will enhance not only RF fingerprinting, but also other wireless signal classification tasks in practical deployments beyond controlled laboratory tests.

VIII. Acknowledgment

This work is supported by the NSF grant CCF-1937500.

References

[1] J. Yu, A. Hu, G. Li, and L. Peng, “A robust RF fingerprinting approach using multisampling convolutional neural network,” *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6786–6799, 2019.
[2] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury, “ORACLE: Optimized radio classification through convolutional neural networks,” in *IEEE INFOCOM 2019*, pp. 370–378.
[3] K. Sankhe, M. Belgiovine, F. Zhou, L. Angioloni, F. Restuccia, S. D’Oro, T. Melodia, S. Ioannidis, and K. Chowdhury, “No radio left behind: Radio fingerprinting through deep learning of physical-layer hardware impairments,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 165–178, 2019.
[4] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.
[5] S. J. Pan, V. W. Zheng, Q. Yang, and D. H. Hu, “Transfer learning for wifi-based indoor localization,” in *Association for the advancement of artificial intelligence (AAAI) workshop*, vol. 6. The Association for the Advancement of Artificial Intelligence Palo Alto, 2008.
[6] T. S. Nazaré, G. B. P. da Costa, W. A. Contato, and M. Ponti, “Deep convolutional neural networks and noisy images,” in *Iberoamerican Congress on Pattern Recognition*. Springer, 2017, pp. 416–424.
[7] F. Restuccia, S. D’Oro, A. Al-Shawabka, M. Belgiovine, L. Angioloni, S. Ioannidis, K. Chowdhury, and T. Melodia, “DeepRadioID: Real-time channel-resilient optimization of deep learning-based radio fingerprinting algorithms,” in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2019, pp. 51–60.

[8] B. Tang, Y. Tu, Z. Zhang, and Y. Lin, “Digital signal modulation classification with data augmentation using generative adversarial nets in cognitive radio networks,” *IEEE Access*, vol. 6, pp. 15 713–15 722, 2018.

[9] Genesys Lab, “Genesys lab ML datasets,” http://genesys-lab.org/mldatasets.

[10] T. Jian, B. C. Rendon, E. Ojuba, N. Soltani, Z. Wang, K. Sankhe, A. Gritsenko, J. Dy, K. Chowdhury, and S. Ioannidis, “Deep Learning for RF Fingerprinting: A Massive Experimental Study,” *IEEE Internet of Things Magazine*, vol. 3, no. 1, pp. 50–57, 2020.

[11] N. Soltani, K. Sankhe, S. Ioannidis, D. Jaisinghani, and K. Chowdhury, “Spectrum Awareness at the Edge: Modulation Classification using Smartphones,” in *IEEE DySPAN 2019*, pp. 1–10.

[12] A. Alimohammad, S. F. Fard, and B. F. Cockburn, “Filter-based fading channel modeling,” *Modelling and Simulation in Engineering*, vol. 2012, 2012.

[13] V. Erceg, “IEEE P802.11 wireless LANs TGn channel models,” *IEEE 802.11-03/940r4*, 2004.

Nasim Soltani is a PhD student at the department of Electrical and Computer Engineering of Northeastern University, Boston. Her current research focuses on deep learning algorithms for signal classification.

Kunal Sankhe is currently pursuing the Ph.D. degree in computer engineering with Northeastern University, Boston. His research interests are implementing deep learning in wireless domain and developing a cross-layer communication framework for the IoT.

Jennifer Dy is a Professor at the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA. Her research spans both fundamental research in machine learning and their application to biomedical imaging, health, science and engineering, with contributions in unsupervised learning, dimensionality reduction, feature selection, learning from uncertain experts, active learning, Bayesian models, and deep representations.

Stratis Ioannidis is an Associate Professor in the Electrical and Computer Engineering Department of Northeastern University, in Boston, MA, where he also holds a courtesy appointment with the Khoury College of Computer Sciences. His research interests span machine learning, distributed systems, networking, optimization, and privacy.

Kaushik Chowdhury is a Professor at Northeastern University, Boston, MA. His research interests involve systems aspects of networked robotics, machine learning for wireless communications and networking, wireless energy transfer, and large-scale experimental deployment of emerging wireless technologies.