Deep Learning for RF Fingerprinting: A Massive Experimental Study

Tong Jian, Bruno Costa Rendon, Emmanuel Ojuba, Nasim Soltani, Zifeng Wang, Kunal Sankhe, Andrey Gritsenko, Jennifer Dy, Kaushik Chowdhury, and Stratis Ioannidis

Abstract

RF fingerprinting is a key security mechanism that allows device identification by learning unchanging, hardware-based characteristics of the transmitter. In this article, we demonstrate how machine learning techniques impact RF fingerprinting by analyzing a dataset of 400 GB of in-phase (I) and quadrature (Q) signal data transmitted by 10,000 radios. Our deep convolutional neural network architectures take raw and processed IQ samples as input to identify devices under a variety of practical conditions, including changing channels, noise levels, training data sizes, and computational overheads, among others. The contributions of the article are as follows: (i) This is the first work, to the best of our knowledge, reporting on RF fingerprinting and scalability issues on very large device populations, in the range of 50–10,000 devices. (ii) We investigate how to mitigate the effect of the wireless channel through a feature engineering step that we refer to as *partial equalization*. (iii) We provide a comprehensive performance evaluation of both a custom-designed and a modified ResNet architecture, with insights on which one may be preferred under specific environmental conditions.

Introduction

The IoT revolution is underway; according to recent surveys, this market segment is poised to become a multi-trillion-dollar industry within the next year. Furthermore, over 200 billion distinct IoT devices will be actively deployed in the next five years. Thus, designing scalable, accurate, energy-efficient, and tamper-proof authentication mechanisms has now become more important than ever. While great strides have been made in cryptography and other cross-protocol layer approaches to detect malicious activity, it is unclear how these methods:

- May scale down to extremely resource-constrained IoT devices with minimal processing ability
- Offer resilience to increasingly sophisticated tools that can "take over" nodes via software attacks, spoofed IDs, and so on

Instead, we propose an approach to RF fingerprinting that classifies a device based on unique transmitter-side electronic circuit characteristics that impart a distinct "signature" to any signal emitted by it. Since these subtle modifications to the signal occur naturally without any intentional software processing, RF fingerprinting is a promising solution for authenticating low-power/computationally capable IoT devices.

This article explores the limits of RF fingerprinting using machine learning (ML) by analyzing a dataset of 400 GB of signal traces from over 10,000 radio transmitters. The key idea here is to design deep convolutional neural network (CNN) architectures that exploit unique and unchangeable features embedded in the transmitted signals that are analyzed through the in-phase (I) and quadrature (Q) components. These modulations include information about IQ imbalance, phase noise, and carrier frequency offset, all of which impart a unique *signature* to the transmitting device [1]. The concept of RF fingerprinting itself has been previously used for authentication [2–4]. However, the impact of the channel, signal-to-noise ratio (SNR), number of devices, training dataset size, and computational time, among others, has not been systematically analyzed. This article attempts to bridge this gap, using a dataset composed of IQ components of received signals collected from two different wireless standards. The first standard is commercial off-the-shelf (COTS) 802.11b/g/n WiFi, and the second type is the Automatic Dependent Surveillance-Broadcast (ADS-B) standard used in airplane status updates.

Digital Object Identifier: 10.1109/IOTM.0001.1900065

Need for ML for RF Fingerprinting

Traditional RF fingerprinting relies on complex and protocol-specific feature extraction techniques [5, 6] that require domain knowledge. Moreover, each emerging wireless standard results in new packet frame structures, channel bandwidths, modulation choices, and coding schemes, which:

- Are often incompatible with previous standards
- Require a redesign of fingerprinting algorithms and, possibly, retraining with new datasets

Our goal here is to explore ML techniques that are robust to software changes, and instead rely on hardware-specific features of transmitters that remain the same even if the communication protocol changes over time. Many of these features introduce only subtle changes in the signal, which can easily be lost in ambient noise and random channel-induced perturbations.

We choose CNNs for our fingerprinting task for several reasons. Compared to the traditional ML techniques (e.g., SVM, KNN, and Random Forest), deep architectures such as CNNs act better as feature interpreters. CNN architectures typically consist of numerous neural network layers, each trained to detect and capture salient, discriminative data features. This property perfectly matches RF fingerprinting setting -- the latent device information over IQ samples will be encoded into features via training, and finally contribute to device identification.

Beyond this, compared to other deep architectures, CNNs have a demonstrated performance record on numerous inference problems across application domains. Employing CNNs allows us to leverage a broad array of well-known architectures that have contributed to this record [7–10]. As an additional advantage, signals that are discriminative for RF fingerprinting tasks may appear at arbitrary positions in a variable length transmission (i.e., an IQ sequence). As a result, classifiers need to be *shift-invariant*, detecting discriminative features irrespective of where they occur. CNNs with convolution and max pooling layers are excellent candidates for shift invariant classification in this setting: convolved weights in each layer detect signals in arbitrary positions in the sequence, and max-pool layers pass the presence of a signal to a higher layer irrespective of where it occurs. Finally, CNN training is data-driven and is thereby a suitable alternative to hand-crafted/protocol-specific techniques, especially to fingerprint transmissions generated by varied sources and protocols.

 2576-3180/20/$25.00 © 2020 IEEE

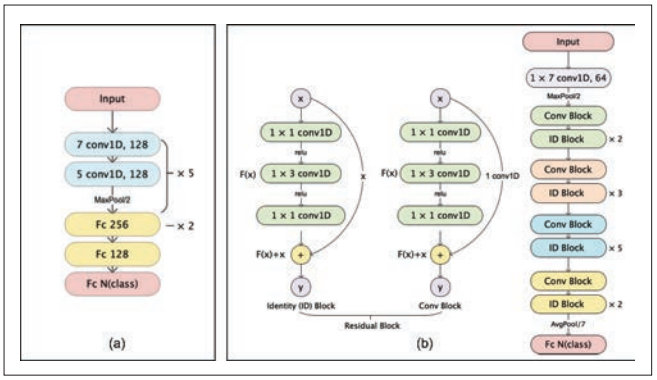


FIGURE 1. 1 Proposed deep CNN architectures: a) baseline model; b) ResNet-50-1D.

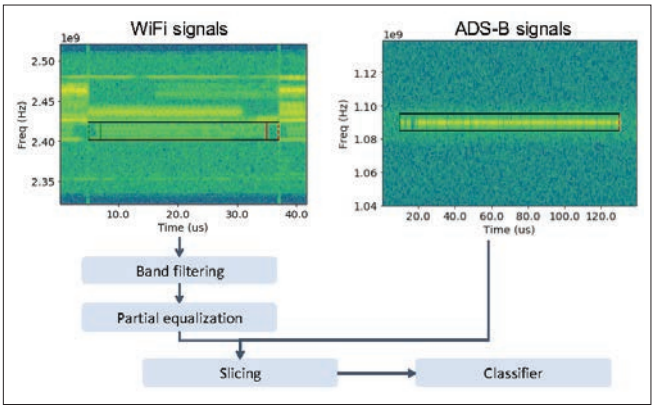


FIGURE 2. Pre-processing and classification pipeline. For WiFi transmissions (left), we perform band filtering (i.e., extracting the signal shown by the bounding box), and partial equalization. Filtering moves the signal to the baseband and applies a low pass filter. Partial equalization on filtered WiFi signal removes only the effect of the channel from received IQ samples, without removing other device-specific imperfections. The sliding window creates multiple slices that are fed to the classifier for training and testing.

Deep CNN Architectures

Popular CNN models include AlexNet [7], GoogleNet [9], VGG16 [8], and ResNet [10]. Although we experimented with all four architectures, we report here two deep CNN architectures: a custom-designed *baseline* model inspired by Alex-Net, and ResNet-50-1D, an architecture based on ResNet, which has proved to be remarkably successful in many domains. The baseline model is a modified version of Alex-Net, adapted for RF fingerprinting tasks. As shown in Fig. 1a, it contains 10 convolution layers and five max pooling layers, organized in five stacks. These are followed by four fully connected layers. ResNet-50-1D, shown in Fig. 1b, is a modified version of ResNet. We describe this modification in detail in the "Classification with Deep CNN Architectures" section. In general, the more layers in the network, the richer the features that it extracts. However, as the network depth increases, the accuracy during training may saturate and even decrease rapidly. This phenomenon is referred to as the *vanishing gradient* effect [11]. ResNet introduces a residual block that allows the creation of deeper network architectures by carrying forward the input of every block to its output. This mitigates the vanishing gradient effect, which is the main cause of accuracy loss in deeper architectures.

Through this article and the accompanying public release of code, we aim to provide architectures with a fine-tuned set of hyperparameters and usage guidelines (i.e., which architecture to use when) to accelerate CNN-enabled research in the wireless community.

We make the following key contributions:

- To the best of our knowledge, we report the first work on RF fingerprinting and scalability issues on very large device populations, in the range of 50–10,000 devices.
- We provide a comprehensive performance evaluation of the baseline and ResNet-50-1D architectures with insights on when one of the two should be chosen. A non-intuitive result is that a deeper architecture is not always better.
- We investigate the effect of the wireless channel through a feature engineering step that we refer to as *partial equalization*. Interestingly, experimental results reveal that removing the channel before training does not always give better results.
- We report the computational overhead of training and classification on a graphic processing unit (GPU) platform, under different networks and training data. We make our code publicly available[1] to accelerate community contributions in this exciting topic.

The rest of this article is organized as follows. We first describe in detail the dataset, followed by a pre-processing pipeline for training the CNN classifiers. We then describe the baseline and ResNet-50-1D architectures and explain how they are adapted for use in the RF domain. We then experimentally validate the accuracy and efficiency of the optimized CNN for a number of environmental scenarios. Finally, we conclude with a discussion on open research issues.

Datasets and Pre-Processing Input Features

We first describe the dataset generated by wireless transmissions from over 10,000 devices captured in the wild. We also describe our training pipeline and the different pre-processing steps that we apply to data prior to classification.

Dataset Description

The WiFi dataset contains 5117 devices and 166 transmissions on average for each device. A spectrum analyzer is used to record each transmission, operating at center frequency of either 2.4 GHz or 5.8 GHz with sampling rate 200 MS/s. Each recording consists on average of 18,686 IQ samples. The ADS-B dataset contains 5000 devices and 76 transmissions on average for each device. Each transmission is recorded at the center frequency of 1.09 GHz with sampling rate 100 MS/s. Each recording on average consists of 9156 IQ samples. Each recording file containing consecutive IQ samples is associated with a corresponding metadata file, including the device ID, the transmission frequency, and the protocol used.

The goal of the deep CNN classifier is to predict the device label of each recording. We follow the traditional ML approach of partitioning the dataset into a training set used for training the CNN and a non-overlapping test set for evaluating its prediction quality. We refer to the recordings on both training and test data as *transmission*. Each transmission is a sequence of IQ samples. The transmission *label* is the identity of the transmitting device.

Our pipeline is illustrated in Fig. 2. We first pre-process the dataset. For the WiFi dataset, given an entire transmission, we perform two operations: band filtering and partial equalization. For the ADS-B dataset, we do not perform filtering or equalization, and use only raw IQ samples. This is for the following two reasons:

- ADS-B uses pulse position modulation, resulting in signals of a certain number of pulses.
- Each transmission happens in isolation, with no interfering device.

[1] https://github.com/neu-spiral/RFMLS-NEU}}}

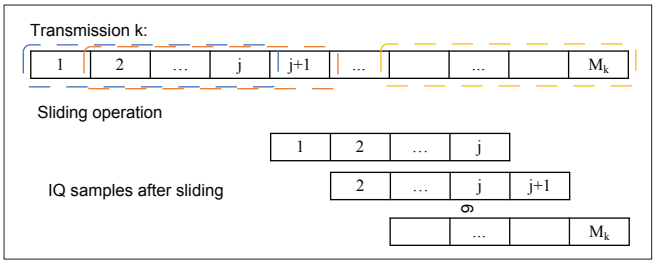


FIGURE 3. An illustration of generating slices from an arbitrary-length sequence. Given a time-series of length M_k, we create M_{k-j+1} subsequences of length j by sliding a window of length j over the larger sequence (or stream) of I/Q samples (with stride 1). This leads to inputs of a fixed length, but also enhances shift invariance. The (full) sequence can be classified by using a label aggregation method over its constituent slices.

We should provide an explanation as to why this difference in preprocessing. For both the WiFi and ADS-B datasets, we apply a sliding window approach to transform the processed signals into a fixed-length form, which the CNN can ingest and classify. Finally, we pass these to the CNN classifier, which outputs a device label prediction. We discuss these pre-processing stages in detail below.

Feature Engineering and Data Pre-Processing

We discuss next the details of the three data pre-processing stages of our pipeline: band filtering, partial equalization, and slicing. The latter is applied to both WiFi and ADS-B signals, while the former two are only applied to WiFi signals.

Band Filtering: WiFi signals in our dataset are collected in a complex, multi-device environment: multiple devices transmit simultaneously on multiple bands. We extract signals for a specific device from the WiFi band it transmits in via band filtering. Taking a signal at a center frequency of 2.4 GHz, for instance, as shown in Fig. 2, we move it to a baseband and apply a low pass filter. This action removes any interference or noise generated out of band.

Partial Equalization: We perform partial equalization on filtered WiFi signals, which aims to remove only an effect of the channel from raw IQ samples without removing the device's imperfections. In particular, we improve the classifier's accuracy by compensating the effect of the channel by channel estimation and equalization, however, leaving the *frequency and sampling offsets* in the IQ samples. Thus, we:

1. First estimate and compensate the carrier and sampling frequency offsets
2. Estimate the channel using the pilot training sequence
3. Reapply the offsets computed in step 1 to obtain our final sequence of equalized symbols, which is fed to the CNN

Slicing: CNNs are restricted to receive signals of a predetermined length as inputs. Recall that transmissions are variable-length time series with two real values (I and Q) per time step. We use a sliding window approach to cut each transmission (in both training and test set) into slices, as shown in Fig. 3. Given an entire transmission k of length M_k, we generate M_{k-j+1} slices of length j by sliding the slicing window with stride 1. Each slice is then labeled individually, receiving the ID of the device that generated the transmission as a label.

In practice, we only generate a random fraction of all possible slices, with the fraction being a design parameter κ, ranging from 1 to j. We select $M_k/j\ \kappa$ slices uniformly at random from each transmission: intuitively, this implies that an IQ sample appears in approximately $\kappa \in [0, j]$ slices. We therefore refer to κ as the *replication factor*. Slicing and randomization have several advantages. First, they ensure we can train our deep CNN on fixed input size — namely, the slice length. Second, they enhance the shift invariance of the features learned by the network: RF imperfections can manifest anywhere in a slice, and randomizing the slice origin enforces this invariance in the data. Third, inferred slice labels on the test set can naturally be aggregated to label an entire transmission, using, for example, the majority label across all its constituent slices. We use a more nuanced aggregation method, which we describe below. Finally, slicing allows us to avoid varying length input issues like vanishing gradients.

Classification with Deep CNN Architectures

The outstanding performance of CNNs [7, 9, 10] in the field of computer vision and natural language processing motivates our investigation on transferring those outcomes to RF fingerprinting. We first describe the deep CNN architectures that we apply to this task. We then describe how we predict a transmission.

Deep CNN Architectures

A convolution layer is the most important component of a CNN. It has a number of different kernels, with the same size; these kernels are convolved with each area across the input with a certain stride, which controls how far a kernel should move across the input. In our case, we use 128 kernels of size 1×7 for the first convolution layer and 128 kernels of size 1×5 for the second convolution layer of each stack. Each kernel learns a variation in time over the I and Q dimension jointly. The convolution layer is usually followed by a predefined activation function that introduces a nonlinearity on feature maps. In our setting, we choose rectified linear units (ReLU), one of the most popular activation functions. Compared to other activation functions, ReLU sets negative values to zero, which results in sparser outputs. The sparsity may be pursued for multiple reasons in a CNN, but it mainly provides robustness to small changes in input, such as noise.

The pooling layer downsamples the input, reducing parameters as well as the computational burden. Typically, the input is divided into several sub-areas, and the output is computed by aggregation of all values in each pool. The most commonly used aggregation is max pooling, which outputs the maximum value of a pool. In our case, we set the pool size to two; pools do not overlap.

Fully connected (or dense) layers, whose neurons are fully connected to the previous layer, are often used after convolution and pooling layers. The dense layer generates high-level features. The baseline model has two dense layers of 256 neurons, one dense layer of 128 neurons, and one classifier layer of outputs with size equal to the number of classes. We use the softmax function as an activation in the classifier layer, and categorical cross-entropy as loss function. Thus, the output of the network is the probability of each class.

There are different types of ResNet architectures, such as ResNet 18, 34, 50, 101, and 152, named after the total number of convolution and fully connected layers. Increasing depth leads to significant accuracy gains but also to slower convergence. To balance the trade-off between speed and accuracy, we use a 50-layer ResNet. Moreover, the original ResNet uses two-dimensional convolution layers. To adapt to transmissions that are time series with two features (I and Q) per time step, we use one-dimensional convolution layers (i.e., temporal convolution) instead. We call the resulting architecture ResNet-50-1D.

Transmission Prediction

We describe below how to use the trained CNN to detect the device of a slice as well as of an entire transmission.

We predict the device of a slice as follows. Given a transmission from a test set, we cut it into slices and run the pre-

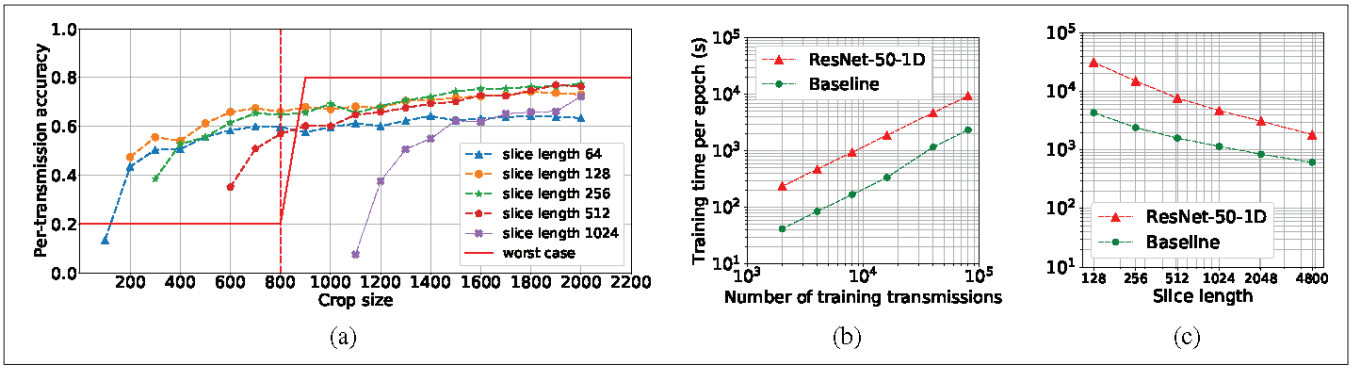


FIGURE 4. a) Device ID analysis. In ADS-B transmissions, the device ID appears after the first 800 samples. To demonstrate that we are not learning the device ID, we crop ADS-B transmissions according to a crop size ranging from 64 to 1024. Then we train and test our baseline model on cropped transmissions. We observe that our model performs well even when not exposed to the device ID (crop size < 800); moreover, including the device ID (crop size > 800) does not significantly affect prediction performance. A "worst case" scenario, in which accuracy increases sharply past 800 samples, is included in this figure for illustrative purposes; b–c) training time analysis. To assess training time dependence on the number of training transmissions (from 2000 to 80,000) and slice length (from 128 to 4800), we vary along one variable at a time and observe its effect on the training time. Experiments illustrate that the baseline model is around six times more efficient than ResNet-50-1D, and both the number of transmissions and slice length have a linear effect on time.

trained classifier on each slice. The output is a vector of probabilities for each device. We predict the device of a slice by taking the most likely device, that is, the one with the highest probability.

We predict the device that generated the entire transmission as follows. Suppose there are N devices, transmission k has n_k slices, and p_{ij} is the outcome of the network indicating the probability of slice j classified as belonging to device i. For each device (e.g., device i), we compute the expected number of slices generated by this device. Formally, this equals $\sum_{j=1}^{n_k} p_{ij}$. We declare the transmission to be the device $\hat{i}$ for which this quantity is maximized: $\hat{i} = \arg\max_i \sum_{j=1}^{n_k} p_{ij}$.

Experimental Setup

Experiment Tasks

We separate the 10,117-device WiFi/ADS-B dataset, described in the "Dataset Description" section, into several subsets. Each corresponds to a different learning task, summarized in Table 1. Each task quantifies the performance of our classifiers under different environmental scenarios, including the impact of the channel, SNR, number of devices, and training dataset size. We describe each task in detail below.

Scalability: Task 1 measures network capabilities with respect to scaling the number of devices the model needs to classify. Each subtask contains both protocols (WiFi and ADS-B); both datasets have the same number of devices, with subtasks within this group having radio populations from 50 to 5000.

Multiburst: Task 1M is identical to Task 1 (1A–1D) with the following modification in the test set: Five transmissions by the same device are joined together, creating a new transmission, which we call a *multiburst*. This task aims to evaluate a more realistic scenario where each device produces multiple transmissions that are to be classified jointly. As the training set is the same, the models trained over training datasets of 1A–1D are tested on this task.

Training set population: Task 2 measures the effect of the training set size on model accuracy. The number of training transmissions per device differs from 2C to 2A, ranging from 63 to 501.

Channel effects: Task 3 assesses the effect of environment conditions on classifier accuracy, using a dataset of 310,000 WiFi transmissions generated by 350 devices. Each subtask aims to simulate a different environmental condition. In subtask 3A, the training and the test set are collected in two different days. This aims to capture variability in channel conditions. We note that this effect is already present in other tasks, as training and test transmissions may be separated by several minutes or hours, and channel conditions change within milliseconds. Nevertheless, subtask 3A makes this difference starker. Subtask 3B is, in contrast, training and testing on the same mixture of two days. Subtasks 3C and 3D contain transmissions transmitted indoors and outdoors, respectively.

SNR: Task 4 directly measures the SNR effect on classifier accuracy. It includes a dataset of 120,000 ADS-B transmissions generated by 100 devices, encountering three different levels of SNR: high (15.3 to 5.1 dB), medium (5.0 to 2.0 dB), and low (1.9 to –13.3 dB).

Bitwise identical: The medium access control (MAC) ID, which is written in each transmission, can be denoted as an additional feature to CNNs. Task 5 assesses whether the CNNs are learning to decode the MAC ID. The task contains 19 nominally identical devices transmitting the same signal using the same MAC ID.

Results and Discussion

We evaluate both baseline and ResNet-50-1D models in terms of per-slice and per-transmission accuracy, as summarized in Table 2. We also report efficiency analysis.

Methodology and Performance Metrics

We implement the baseline and ResNet-50-1D models in Keras with TensorFlow backend on an NVIDIA Cuda enabled Tesla V100 GPU. For the WiFi dataset, we try both filtered and equalized signals as input; for the ADS-B dataset, we use only raw signals.

For each subtask, we separate 20 percent of the training set, and use it as a validation set to determine hyperparameters. In all cases, we optimize several hyperparameters, including slice length, κ, and learning rate, using the validation set. We then use optimized parameters over the entire training set and report accuracy performance on the test set.

To measure the performance of the classifier, we first report per-slice accuracy, which is the fraction of correctly predicted slices over the total number of slices. The second metric is per-transmission accuracy. Recall that we predict the label of a transmission aggregating predictions on its constituent slices using the methodology discussed inthe " Transmission Prediction" section. Per-transmission accuracy can be computed as the fraction of correctly predicted transmissions over the total number of transmissions.

Task	Description	Number of (#) devices	# Train transmissions per device	# Test transmissions per device	Protocol				
					WiFi		ADS-B		
					# Devices	Days	# Devices	Train SNR	Test SNR
1A	Population very high	10,000	22	6	5000	1	5000	>5	>5
1B	Population high	1000	141	55	500	1	500	>5	>5
1C	Population medium	500	141	55	250	1	250	>5	>5
1D	Population low	100	141	55	50	1	50	>5	>5
1MA	Population very high (multiburst)	10,000	22	6	5000	1	5000	>5	>5
1MB	Population high (multiburst)	1000	141	100	500	1	500	>5	>5
1MC	Population medium (multiburst)	500	141	100	250	1	250	>5	>5
1MD	Population low (multiburst)	100	141	100	50	1	50	>5	>5
2A	Training transmissions high	100	501	125	50	1	50	>5	>5
2B	Training transmissions medium	100	313	313	50	1	50	>5	>5
2C	Training transmissions medium	100	63	563	50	1	50	>5	>5
3A	Train one day test another	50	100	100	50	2	–	–	–
3B	Train on a mix of days test on a mix	100	800	200	100	2	–	–	–
3C	Train/test on a single indoor day	100	800	200	100	Indoor	–	–	–
3D	Train/test on a single outdoor day	100	800	200	100	Outdoor	–	–	–
4A	SNR:train high test medium	100	401	401	–	–	100	>5	2–5
4B	SNR:train high test low	100	401	401	–	–	100	>5	<2
4C	SNR:train medium test high	100	401	401	–	–	100	2-5	>5
4D	SNR:train medium test low	100	401	401	–	–	100	2-5	<2
4E	SNR:train low test high	100	401	401	–	–	100	<2	>5
4F	SNR:train low test medium	100	401	401	–	–	100	<2	2-5
5	Bitwise identical	19	8953	2238	19	–	–	–	–

TABLE 1. Overview of RF experiments. Task 1 measures network capabilities with respect to scaling. Task 1M is identical to task 1, with the following modification in the test set: Five transmissions by the same device are joined together, creating a new transmission, resulting in 100 created transmissions in the test set. Task 2 measures the effect of the training set size on model accuracy. Tasks 3 and 4 measure the effect of environmental and channel conditions on model accuracy, using WiFi and ADS-B transmissions, respectively. Finally, task 5 contains 19 bitwise identical devices, transmitting the same signal, using the same MAC ID.

Classification Performance

We draw several conclusions from the performance reported in Table 2.

Scalability: In general, for each subtask in task 1, both baseline and ResNet-50-1D scale gracefully with respect to the total number of devices. For the WiFi dataset, ResNet-50-1D performs well over filtered data, while the baseline architecture performs better over equalized data. For the ADS-B dataset, ResNet-50-1D performs extremely well in predicting a larger number of classes, attaining 0.77 and 0.90 accuracy over 5000 and 500 devices, respectively, while the baseline architecture outperforms ResNet-50-1D over fewer classes, attaining 0.88 and 0.92 accuracy over 250 and 50 devices, respectively.

Multiburst: Task 1M indicates that multiburst can be classified with very high accuracy by both architectures. In the case of subtask 1MA, multiburst classification accuracy for the population of 5000 devices is 0.62 on filtered WiFi (an increase of 135.11 percent over the single burst case) and 0.92 on the raw ADS-B dataset (an increase of 91.7 percent over the single burst case).

Training set population: Task 2 indicates that model accuracy improves by adding more transmissions in the training set. We observe this for both WiFi and ADS-B datasets. Moreover, the baseline model achieves better performance on both protocols than ResNet-50-1D, which indicates that deeper is not always better.

Channel effects: Task 3 shows that environmental conditions indeed affect classifier accuracy. The different-day experiment of subtask 3A illustrates that changes over the channel can severely hamper performance. Interestingly, equalizing with WiFi transmissions ameliorates the impact on performance. Indeed, we attain 0.34 and 0.26 accuracy under equalization for baseline and ResNet-50-1D, respectively; in contrast, prediction over filtered data is close to random guessing. We observe the opposite effect in subtask 3B, in which multiple days appear on both training and test sets.

SNR: An interesting phenomenon appears in task 4 with respect to the effect of SNR. When the model is trained with high-quality data (high SNR) but tested on lower-quality data (low SNR), as seen in subtask 4B, the accuracy is low; when the model is trained with low SNR but tested on high SNR, in subtask 4E, the accuracy is high. The phenomenon is even more pronounced on subtask 4C (trained with medium SNR and tested on high SNR) and 4F (trained with low SNR and tested on medium SNR), where we observe very high accuracy (0.92 and 0.93, respectively). In general, adding noise to the training data makes the network more robust, that is, less affected by the presence of noise in the test set. When the model is trained with high SNR but tested on low SNR, as seen in subtask 4A, the accuracy is low, as in subtasks 4A and 4B. In these cases, the model never sees a good amount of noise in the training set and does not learn how to be invariant and immune to it. In cases where the model is trained with noisy data (medium or low SNR) and then tested on equal or higher-quality data, the performances are comparable if not even better on the test set, such as subtasks 4C, 4E, and 4F. In these cases, the model is trained with noisy data and learns how to ignore it. Additionally, the better performance achieved by the baseline model indicates again that deeper is not always better.

Bitwise similar: Our performance on bitwise similar devices, as well as on the ADS-B dataset, provide a strong indication that our classifier is not learning the device id, a result that we first reported and explored in more detail in [12].

Task	Accuracy on test set Per-slice/Per-transmission accuracy						
	WiFi				ADS-B		
	Filtered		Equalized		Raw		
	Baseline	ResNet-50-1D	Baseline	ResNet-50-1D		Baseline	ResNet-50-1D
1A	0.082/0.130	**0.164/0.262**	0.062/0.101	0.014/0.030		0.374/0.529	**0.574/0.770**
1B	0.299/0.378	**0.393/0.612**	0.327/0.434	0.392/0.555		0.665/0.826	**0.803/0.896**
1C	0.354/0.398	**0.467/0.629**	0.454 /0.478	0.430/0.549		**0.732/0.877**	0.646/0.813
1D	0.335/0.575	**0.490/0.631**	**0.762/0.639**	0.699/0.637		**0.810/0.919**	0.717/0.862
1MA	0.094/0.301	**0.193/0.616**	–	–		0.350/0.829	**0.523/0.918**
1MB	0.272/0.797	**0.375/0.898**	0.334/0.896	0.427/0.799		0.610/0.945	**0.684/0.951**
1MC	0.333/0.740	0.448/0.939	**0.455/0.946**	0.399/0.732		**0.695/0.971**	0.621/0.950
1MD	0.329/0.917	0.478/0.940	0.739/0.975	**0.694/0.990**		**0.796/0.992**	0.706/0.972
2A	0.489/0.684	0.587/0.747	**0.809/0.835**	0.751 0.752		**0.854/0.943**	0.798/0.893
2B	0.459/0.599	0.541/0.682	**0.796/0.756**	0.734/0.710		**0.808/0.895**	0.772/0.852
2C	0.313/0.381	0.363/0.388	**0.606/0.537**	0.566/0.520		**0.687/0.768**	0.565/0.681
3A	0.017/0.016	0.013/0.012	**0.232/0.335**	0.175/0.258	4A	**0.375/0.526**	0.376/0.494
3B	0.444/0.695	**0.520/0.811**	0.678/0.674	0.751/0.735	4B	**115/0.156**	104/0.132
3C	0.457/0.682	**0.574/0.824**	0.0005/0.010	0.0001/0.010	4C	**790/0.916**	720/0.828
3D	0.310/0.598	**0.441/0.746**	0.210/0.432	0.308/0.542	4D	**378/0.600**	333/0.509
5	**0.778/1**	–	0.786/0.957	–	4F	**649/0.925**	594/0.810

TABLE 2. Classification performance, measured in accuracy for different tasks described in Table 1. Both baseline and ResNet-50-1D models scale gracefully on task 1. Task 2 indicates that model accuracy improves by adding more transmissions in the training set. Moreover, environmental and channel conditions (tasks 3 and 4) indeed affect classifier accuracy. Predicting multibursts (task 1M) and bitwise identical devices (task 5) can be performed with extremely high accuracy. In general, accuracies for ADS-B signals are higher than for WiFi signals, indicating thatthey are easier to identify. For WiFi datasets, accuracy for equalized data is not always higher than for filtered data, indicating that removing the channel before training does not always give better results. Finally, the baseline model outperforms ResNet-50-1D in several cases, indicating that deeper is not always better.

On task 5, consisting of a dataset with 19 nominally identical devices, transmitting exactly the same bit sequence, the baseline model achieves 100 percent per-transmission accuracy on filtered WiFi data. This confirms that our shift-invariant model is indeed not learning to decode the MAC ID, and can successfully detect and distinguish unique features introduced by different devices.

In Table 2, classification accuracy for equalized data is sometimes lower than that of raw I/Q samples. The equalization process allows us to remove some channel effects but not all; ideally, we would like to remove environment-dependent effects but not device-specific artifacts. In addition, equalization involves downsampling. The final prediction quality on equalized data depends on a trade-off between:

- Keeping discriminative features
- Not losing information by downsampling
- Removing features that are indeed irrelevant

Hence, prediction performance under equalization is contingent upon how the interplay between these factors manifests in the data, leading to the fluctuations in Table 2.

We conduct additional experiments on ADS-B dataset (task 1D) to demonstrate that we are not learning the device ID. The signal format of ADS-B has a sync pulse that lasts 8 μs (800 samples) followed by either 56 or 112 μs of data. The ID appears in the data portion of the packet. We crop transmissions according to a crop size ranging from 64 to 1024, effectively looking only at the first part of the packet and ignoring the rest. Then we apply our entire pipeline, including slicing, to cropped transmissions in both the training and test sets. This allows us to assess the accuracy of our classification w.r.t. the crop size. This, in turn, allows us to capture whether the model relies on the ID to classify the device. This would be the case if the accuracy remained low for crop size below 800 samples, and increased significantly after 800 samples (indicated in Fig. 4a as "worst case").

As we observed in Fig. 4a, this not the case. The accuracy of the classification increases overall as the function of the crop size as more data becomes available to the classifier. However, irrespective of the slice size used, the accuracy improvement is smooth, and there is no sharp increase past 800 samples. We refer the interested reader to [12] for additional experiments on this topic.

Broader observations: In general, we observe that accuracy for ADS-B signals is higher than for WiFi signals, indicating that they are easier to identify. For WiFi datasets, accuracy for equalized data is not always higher than for filtered data, indicating that removing the channel before training does not always give better results. Finally, the baseline model outperforms ResNet-50-1D in several cases, indicating that deeper is not always better.

Training Time Analysis

The training time of a CNN depends on the network architecture, the number of transmissions, and slice length. Specifically, the CNN size and depth have a direct effect on training time. The number of transmissions and slice length together determine the total number of slices evaluated during training. Intuitively, the longer the slice length, the fewer slices generated for each transmission.

We analyze the training time of both baseline and ResNet-50-1D models based on Task 1D. We vary the number of transmissions (from 2000 to 80,000) and slice length (from 128 to 4800), and observe its effect on the training time. Specifically, we fix κ as 10, and initially set training set size and slice length as 40,000 and 1024, respectively. Experiments are based on a configuration of one Tesla V100 GPU for timing reports, as shown in Figs. 4b and 4c.

In general, we notice that:

- The baseline model is around six times more efficient than ResNet-50-1D.

- Both the number of transmissions and slice length have a linear effect on time.

Open Research Issues

An interesting future direction is to find a network architecture appropriate for even larger-scale classification tasks (involving, e.g., 50,000 devices). Hierarchical classifiers [13] can be applied to this task; such approaches cluster related classes together potentially in a way that increases the discriminative power of the resulting classifiers. Other possible architectures could be variational auto-encoders [14], which exhibits a powerful ability to extract features. Besides network architectures, we can also explore feature engineering to infer waveform characteristics. Finally, channel and environment conditions affect the model performance. Rather than removing channel conditions in the pre-processing stage, adversarial learning [15] can increase robustness against channel variations and thus improve performance on raw signals.

Conclusion

RF fingerprinting is critical to improving wireless communication security. In this article, we comprehensively investigate two CNN architectures for RF fingerprinting under different environmental scenarios, including the impact of the channel, SNR, number of devices, and training dataset size. We have shown that both baseline and ResNet-50-1D models

- Are scalable to very large populations (10,000 devices)
- Can handle different environmental conditions, such as channel effects and SNR
- Perform extremely well on multiburst and bitwise similar tasks, which further indicates network capabilities on identifying unique features introduced by hardware impairments

Acknowledgments

This work is supported by the Defense Advanced Research Projects Agengy (DARPA) under RFMLS program contract N00164-18-R-WQ80. We are grateful to Paul Tilghman, program manager at DARPA, and Esko Jaska for their insightful comments and suggestions.

References

[1] K. Sankhe *et al.*, "ORACLE: Optimized Radio Classification Through Convolutional Neural Networks," *IEEE Int'l. Conf. Comp. Commun.*, 2019.
[2] O. Ureten and N. Serinken, "Wireless Security Through RF Fingerprinting," *Canadian J. Electrical and Comp. Engineering*, vol. 32, no. 1, 2007, pp. 27–33.
[3] W. C. S. II *et al.*, "Radio Frequency Fingerprinting Commercial Communication Devices to Enhance Electronic Security," *Int'l. J. Electronic Security and Digital Forensics*, vol. 1, no. 3, 2008, pp. 301–22.
[4] V. Lakafosis *et al.*, "RF Fingerprinting Physical Objects for Anticounterfeiting applications," *IEEE Trans. Microwave Theory and Techniques*, vol. 59, no. 2, 2011, pp. 504–14.
[5] V. Brik *et al.*, "Wireless Device Identification with Radiometric Signatures," *Proc. 14th ACM Int'l. Conf. Mobile Computing and Networking*, 2008, pp. 116–27.
[6] T. D. Vo-Huu, T. D. Vo-Huu, and G. Noubir, "Fingerprinting Wi-Fi Devices Using Software Defined Radios," *Proc. 9th ACM Conf. Security and Privacy in Wireless and Mobile Networks*, 2016, pp. 3–14.
[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Info. Processing Systems 25*, 2012, pp. 1097–105.
[8] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv preprint arXiv:1409.1556, 2014.
[9] C. Szegedy *et al.*, "Going Deeper with Convolutions," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2015, pp. 1–9.
[10] K. He *et al.*, "Deep Residual Learning for Image Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 770–78.
[11] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," *Proc. 13th Int'l. Conf. Artificial Intelligence and Statistics*, vol. 9, 2010, pp. 249–56.
[12] T. Jian *et al.*, "MAC ID Spoofing-Resistant Radio Fingerprinting," *IEEE GlobalSIP*, 2019.
[13] N. Agarwal, V. N. Balasubramanian, and C. V. Jawahar, "Improving Multiclass Classification by Deep Networks Using DAGSVM and Triplet Loss," *Pattern Recognition Letters*, vol. 112, 2018, pp. 184–90.
[14] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *Proc. Int'l. Conf. Learning Representations*, 2013.
[15] I. Goodfellow *et al.*, "Generative Adversarial Nets," *Advances in Neural Information Processing Systems 27*, 2014, pp. 2672–80.

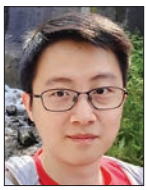
Biographies

Tong Jian is currently pursuing a Ph.D. degree in the Department of Electrical and Computer Engineering, Northeastern University, Boston, Massachusetts. She received her M.Sc. (2016) in electrical engineering from Rensselaer Polytechnic Institute, New York. She works under the guidance of Prof. Stratis Ioannidis in the field of machine learning. Her current research efforts are focused on the application of machine learning in the domain of wireless communication.

Bruno Costa Rendon graduated with a Master's degree in machine learning from the College of Electrical and Computer Engineering at Northeastern University. He is currently working at Raytheon on machine learning projects. He is native to Barcelona, Spain, and is now a local in Boston, Massachusetts. He is passionate about machine learning and computer vision.

Emmanuel Ojuba is currently an M.Sc student in the Department of Electrical and Computer Engineering at Northeastern University under the guidance of Prof Jennifer Dy. He received his B.Sc (2015) in mechanical engineering from Purdue University, Indiana. His current research efforts are focused on the application of deep learning in wireless communications

Nasim Soltani is currently a Ph.D. student in the Department of Electrical and Computer Engineering at Northeastern University. She is pursuing her Ph.D. under the guidance of Prof. Chowdhury in wireless communication. Her current research area focuses on deep learning algorithms for signal classification. She is interested in algorithms and methods for implementing deep learning on resource constrained devices.



Zifeng Wang is currently pursuing a Ph.D. degree in the Department of Electrical and Computer Engineering, Northeastern University. He received his B.Sc. (2014) in electronic engineering from Tsinghua University, China. He works under the guidance of Prof. Jennifer Dy in machine learning. His current research focuses on lifelong learning, representation learning, and the application of machine learning in the domain of biostatistics and wireless communication.

Kunal Sankhe is currently pursuing a Ph.D. degree in computer engineering at Northeastern University under the supervision of Prof. K. Chowdhury. He received his B.Eng. (2011) in electronics and telecommunication engineering from Mumbai University and his M.S. (2016) in electronics and communication engineering from IIIT Hyderabad. His current research efforts are focused on implementing deep learning in wireless domain and developing a cross-layer communication framework for the Internet of Things.

Andrey Gritsenko is a research associate in the Electrical and Computer Engineering Department at Northeastern University. He received his M.Sc. (2010) in applied mathematics and computer science from the Stavropol State University, Russia, and his Ph.D. (2017) in industrial engineering from the University of Iowa. His research interests span the areas of interpretable machine learning, big data, signal processing, and graph mining.

Jennifer Dy is a professor in the Department of Electrical and Computer Engineering, Northeastern University, where she first joined the faculty in 2002. She received her M.S. and Ph.D. in 1997 and 2001, respectively, from Purdue University, and her B.S. degree from the University of the Philippines in 1993. Her research spans both fundamental research in machine learning and their application to biomedical imaging, health, science, and engineering, with research contributions in unsupervised learning, dimensionality reduction, feature selection, learning from uncertain experts, active learning, Bayesian models, and deep representations. She received an NSF Career award in 2004. She has served or is serving as Secretary for the International Machine Learning Society, Associate Editor/Editorial Board member for the *Journal of Machine Learning Research*, *Machine Learning*, and *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Organizing and/or Technical Program Committee member for premier conferences in machine learning and data mining (ICML, NeurIPS, ACM SIGKDD,

AAAI, IJCAI, UAI, AISTATS, SIAM SDM), and Program Co-Chair for SIAM SDM 2013 and ICML 2018.

Kaushik Chowdhury is an associate professor at Northeastern University. He was awarded the Presidential Early Career Award for Scientists and Engineers (PECASE), DARPA Young Faculty Award, and the Office of Naval Research Early Career Award in 2016, and the NSF CAREER in 2015. He received best paper awards at IEEE GLOBECOM '19, IEEE DySPAN '19, IEEE INFOCOM '18, ACM SenSys '18 (runners up), IEEE ICC '09, '12 and '13, and ICNC '13. He is presently a co-director of the Platforms for Advanced Wireless Research (PAWR) project office. His current research interests involve systems aspects of networked robotics, machine learning for agile spectrum sensing/access, wireless energy transfer, and large-scale experimental deployment of emerging wireless technologies.

Stratis Ioannidis is an assistant professor in the Electrical and Computer Engineering Department at Northeastern University, where he also holds a courtesy appointment with the Khoury College of Computer Sciences. He received his B.Sc. (2002) in electrical and computer engineering from the National Technical University of Athens, Greece, and his M.Sc. (2004) and Ph.D. (2009) in computer science from the University of Toronto, Canada. Prior to joining Northeastern, he was a research scientist at the Technicolor research centers in Paris, France, and Palo Alto, California, as well as at Yahoo Labs in Sunnyvale, California. He is the recipient of an NSF CAREER Award, a Google Faculty Research Award, and best paper awards at ACM ICN 2017 and IEEE DySPAN 2019. His research interests span machine learning, distributed systems, networking, optimization, and privacy.