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 level, 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.

I. 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 identification 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 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 paper 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.

A. 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 (i) are often incompatible with previous standards, and (ii) 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 be easily 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 irrespectively 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.

B. Deep CNN Architectures

Popular CNN models include AlexNet [7], GoogleNet [9], VGG16 [8] and ResNet [10]. Though 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 ten 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 the ResNet. We describe this modification in detail in section Classification with Deep CNN Architectures. 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 paper and 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:
• 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 GPU platform, under different networks and training data. We make our code publicly available [12] 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.

II. 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.

A. 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 18686 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: (i) ADS-B uses pulse position modulation, (ii) ADS-B transmission frequency and protocol used.

• 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.

https://github.com/neu-spiral/RFMLS-NEU
resulting in signals of a certain number of pulses, and (ii) each transmission happens in isolation, with no interfering device. 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.

**B. 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 that 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 (i) first estimate and compensate the carrier and sampling frequency offsets, (ii) estimate the channel using the pilot training sequence, and (iii) reapply the offsets computed in step (i) to obtain our final sequence of equalized symbols that 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 of length \( M_k \), we generate \( M_k - j + 1 \) slices of length \( j \) by sliding a window with stride \( j \) over the larger sequence (or stream) of IQ 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.

In practice, we only generate a random fraction of all possible slices, with the fraction being a design parameter \( \kappa \), ranging from 1 to \( j \). We select \( \frac{M_k}{j} \kappa \) 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 \( \kappa \) 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.

**III. CLASSIFICATION WITH DEEP CNN ARCHITECTURES**

The outstanding performance of CNNs 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 an transmission.

**A. 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 \( 1 \times 7 \) for the first convolution layer, and 128 kernels of size \( 1 \times 5 \) 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 non-linearity 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 transmission which 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.

B. 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 test set, we cut it into slices and run the pre-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, take device $i$ for instance, we compute the expected number of slices generated by this device. Formally, this equals to $\sum_{j=1}^{n_k} p_{ij}$. We declare the transmission to be the device $\hat{i}$ for which this quantity is maximized: $\hat{i} = \arg\max_{i} \sum_{j=1}^{n_k} p_{ij}$.

IV. EXPERIMENTAL SETUP

A. Experiment Tasks

We separate the 10117-device WiFi/ADS-B dataset, described in section Dataset Description, into several subsets.
the baseline model outperforms ResNet-50-1D in several cases, indicating deeper is not always better. For WiFi datasets, accuracy (Task 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 they are easier to identify.

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. Task 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.

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 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, Task 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.
equalized signals as input; for 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, $\kappa$, and learning rate, using the validation set. We then use optimized parameters over entire training set and report accuracy performance on test set.

To measure the performance of the classifier, we first report per-slice accuracy, which is the fraction of correctly predicted slices over 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 in section Transmission Prediction. Per-transmission accuracy can be computed as the fraction of correctly predicted transmissions over total number of transmissions.

### B. Classification Performance

We draw several conclusions from the performance reported on 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 on predicting 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 5,000 devices is 0.62 on filtered WiFi (an increase by 135.11% over the single burst case), and 0.92 on the raw ADS-B dataset (an increase by 91.7% 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.

**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.

**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]. 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 IQ 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 tradeoff between (a) keeping discriminative features, (b) not losing information by downsampling, and (c) 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 VII.

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 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 on Fig. 4 as “worst case”).

As we observed in Fig. 4, 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, irrespectively 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 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 deeper is not always better.

C. 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 is, 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 $\kappa$ as 10, and as an initial, 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 Table VII.

In general, we notice that (i) the baseline model is around six times more efficient than ResNet-50-1D, (ii) both the number of transmissions and slice length have a linear effect on time.

VI. OPEN RESEARCH ISSUES

An interesting future direction is to find a network architecture that 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 pre-processing stage, adversarial learning [13] can increase robustness against channel variations and thus improve performance on raw signals.

VII. 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 (i) are scalable to very large populations (10,000 devices), (ii) can handle different environmental conditions, such as channel effects and SNR, and (iii) perform extremely well on multiburst and bitwise similar tasks, which further indicates network capabilities on identifying unique features introduced by hardware impairments.

VIII. ACKNOWLEDGEMENT

This work is supported by the Defense Advanced Research Projects Agency (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


**Tong Jian** is currently pursuing the Ph.D. degree in the Department of Electrical and Computer Engineering, Northeastern University. She received her M.Sc. (2016) in Electrical Engineering from Rensselaer Polytechnic Institute, NY. 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 Masters degree in Machine Learning from the College of Electrical and Computer Engineering at Northeastern University. Bruno is currently working at Raytheon on machine learning projects. He is native to Barcelona, Spain, and is now a local in Boston. 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, IN. His current research efforts are focused on the application of deep learning in wireless communications.
Jennifer Dy is Professor at the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, 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 journal, 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 Associate Professor at Northeastern University, Boston, MA. He was awarded the Presidential Early Career Award for Scientists and Engineers (PECASE), DARPA Young Faculty Award, the Office of Naval Research Early Career Award in 2016, and the NSF CAREER in 2015. He received best paper awards at IEEE GLOBECOM19, IEEE DySPAN’19, IEEE INFOCOM18, ACM SenSys18 (runners up), IEEE ICC09, 12 and 13, and ICNC13. 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 of Northeastern University, in Boston, MA, 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, CA, as well as at Yahoo Labs in Sunnyvale, CA. He is the recipient of an NSF CAREER Award, a Google Faculty Research Award, and a best paper award at ACM ICN 2017 and IEEE DySPAN 2019. His research interests span machine learning, distributed systems, networking, optimization, and privacy.
Figure 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 apply 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.
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.
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.