Hazra, S., Voigt, T. & Yan, W. (4-7 October, 2021). *PLIO: Physical Layer Identification using One-shot Learning* [Proceedings]. 2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS), Denver, CO, USA.

DOI: 10.1109/MASS52906.2021.00050

PLIO: Physical Layer Identification using One-shot Learning

Saptarshi Hazra
Research Institutes of Sweden
Stockholm, Sweden
saptarshi.hazra@ri.se

Thiemo Voigt
Research Institutes of Sweden
Stockholm, Sweden
thiemo.voigt@ri.se

Uppsala University
Uppsala, Sweden
thiemo.voigt@it.uu.se

Wenqing Yan
Uppsala University
Uppsala, Sweden
wenqing.yan@it.uu.se

***Abstract*—The Internet of Things (IoT) is connecting a massive scale of everyday objects to the internet. We need to ensure the secure connectivity and authentication of these devices. Physical (PHY)-layer identification methods can distinguish between different devices by leveraging their unique hardware imperfections. But these methods typically require large quantities of training data which makes them impractical for large deployment scenarios. Also, these methods do not address the PHY-layer identification of new devices joining an IoT network. In this paper, we propose a PHY-layer identification method using one-shot learning that can identify new devices using the network solicitation packet of the devices as reference packets. We show that our method can accurately identify new devices without training, achieving a precision and recall over 80% even in the presence of 10 dBm noise. Furthermore, we show that with minimal retraining using only three packets from each device, we can accurately identify all devices in the IoT network with a precision and recall of 93%.**

***Index Terms*—RF Fingerprinting, Security, Deep-Learning**

I. Introduction

A wide-variety of applications such as healthcare [21], smart cities [13], and smart industries [5] rely on Internet of Things (IoT) devices. While typically such devices are secured using digital certificates, their low-power nature makes it impossible to use complex cryptographic functions for device authentication. A rogue device can disguise itself as a trusted device by stealing the cryptographic authentication keys from a trusted device. As an added security mechanism against such impersonation attacks, Physical (PHY)-layer identification can be used to authenticate these IoT devices without imposing additional energy costs to the IoT devices themselves. PHY-layer identification uses the minute differences in the transmitted analog signal to identify a device. These differences are caused by hardware imperfections in the analog components of a radio transmitter.

The existing work to identify devices based on their transmitted signal fall mainly into two categories. Technology-specific methods use protocol-specific handcrafted features for a particular wireless access technology to identify devices. High protocol dependence makes these methods hard to adapt to other wireless access technologies. Technology-agnostic methods use deep neural networks as feature extractors to automatically extract device-specific features using radio signals. Technology-agnostic methods are more desirable in IoT networks as IoT networks typically consist of heterogeneous wireless access technologies.

Technology-agnostic PHY-layer identification methods are typically used for fixed networks, where the devices in the network are known a priori. A deep neural network uses radio signals to learn the mapping between the characteristics of the radio signal and the device identity. For robust identification of new devices, the neural network needs to be retrained with a large amount of data, typically hundreds of packets from each device [26]. This process is time and computation intensive, which is infeasible for real-world dynamic IoT networks where new devices are added to the network as the demands of the application grow.

Some distinguishing features for the transmitters, such as Center Frequency Offset (CFO) and Sampling Frequency Offset (SFO), are manifested in the received signal with respect to the CFO and SFO of the receiver [10]. Moreover, the receiver adds its characteristics, such as amplifier distortion, antenna characteristics, and I/Q quadrature demodulator characteristics to the received signal. For practical applications, the same PHY-layer identification neural network should be deployable on different receivers.

To simplify the extension of IoT networks and enable PHY-layer identification for new devices, we use a verification neural network [7] based on contrastive learning methods to compare between characteristics of devices. Contrastive-learning methods [12] learn a mapping from a high-dimensional input space to a low-dimensional representation space so that similar inputs are located nearby in the representation space, while dissimilar inputs are located far away. The contrastive mapping can then differentiate between inputs from new devices without retraining the network. Such methods have been shown to be highly effective at generalizing to new classes with few examples [16], [20].

In this paper, we propose PLIO - Physical Layer Identification using One-shot learning. We store a single radio packet from all the devices in an IoT network as our reference packets. PLIO uses a verification neural network, similar to the image identification network proposed by Koch et al. [16]. PLIO compares an input packet to the reference packets for all the devices to identify the source device for that input packet using the verification neural network. This process of identification of new devices based on a single reference packet is called one-shot learning. We categorize the devices

in our network as: *seen* and *unseen* devices. *Seen* devices are used to train our verification neural network. *Unseen* devices are not used to train our verification neural network. PLIO's goal is to provide PHY-layer identification of both *seen* and *unseen* devices.

Contributions:

- We present PLIO, a one-shot learning based PHY-layer identification method, which can identify the *unseen* devices using a set of reference packets with a precision and recall greater than 80% for two separate datasets, in comparison to 48% achieved by previous works.
- We exhibit that PLIO can also perform PHY-layer identification for an existing IoT network with new devices joining the network. PLIO can identify all devices, *seen* and *unseen*, in such scenarios with a precision and recall greater than 75%. We further show that minimal retraining of the neural network can boost the precision and recall for the identification to 93%.
- We show that PLIO can create a single deep neural network model that can adapt to different IoT networks even with a different receiver in different noisy radio environments.

II. Related Work

Although there is a lot of work [2], [3], [14], [15], [29], [30] focusing on PHY-layer identification of *seen* devices, the problem of generalization of PHY-layer identification to *unseen* devices is relatively unexplored.

There are previous works [9], [17], [28] detecting *unseen* traffic classes with bit-level information using deep learning methods. In contrast to their work, we identify individual device based on its hardware characteristics without looking at the content of the packets. Shi et al. [24] address problem of detection of *unseen* signal classes for PHY-layer identification. Their work detects *unseen* signals using the Minimum Covariance Determinant (MCD) method, and k-means clustering algorithm. Gritsenko et al. [11] propose an outlier detection method for detecting packets from *unseen* devices. However, all these works focus on detecting signals from different *unseen* devices as a seperate class. Hence, all *unseen* devices get identified as a single device class. In this paper, we differentiate between *unseen* devices.

Robyns et al. [22] detect *unseen* devices by modeling the output of the softmax layer of their deep neural network as a multivariate Gaussian distribution. Their work further classifies between the *unseen* devices by using unsupervised Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering of the deep neural network representation of an radio symbol. However, their work only discusses the impact of the number of *unseen* devices on identification performance. Furthermore, their method exhibits a recall and precision of only 48% for five *unseen* devices. Wong et al. [27] propose the use of DBSCAN clustering of the deep neural network representation of an input to identify *seen* and *unseen* devices. Their results show that this method can only identify 15 *seen* devices accurately when *unseen* devices are present. Different from these works, we use contrastive-learning methods to design our deep neural network and demonstrate significant improvements in *unseen* device identification. In addition, we explore the adaptability of our method for different radio chipsets, different receivers, and noisy radio conditions.

III. Approach

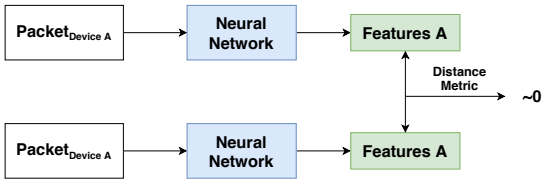
Problem Formulation: We assume a network of N devices, and that we have trained a deep neural network to identify the source of packets originating from these N devices. M new devices can join the network by sending network solicitation and association packets. N and M are the number of *seen* and *unseen* devices in our network. We should not identify the source of a packet from one of the M *unseen* devices as one of the N *seen* devices or any of the other *unseen* devices. We formulate our research question as: Can we generalize the learnings of a deep neural network trained on N *seen* devices such that we can identify the source of a packet from any of the M *unseen* devices or N *seen* devices, using the network solicitation packets as a reference?

A. Overview of PLIO

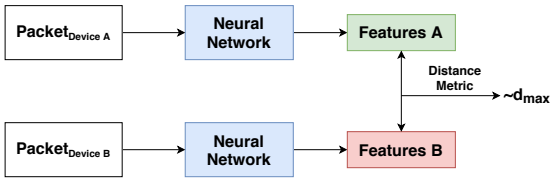
During network association, each device sends its Device Unique Identifier (DUI) using a network solicitation packet to join the network. In this work, we do not consider the security of the network solicitation packet, as the security of the network solicitation packets is handled by higher layer application security protocols. Hence, we assume the network solicitation packets are authentic packets from trusted devices. We store one network solicitation packet from each device as the reference packet. We design a Siamese network to compare an incoming packet with the reference packets for all devices in the IoT network.

Siamese network is a contrastive-learning based deep neural network architecture introduced for signature verification [7]. It consists of twin identical subnetworks, which share the same parameters and weights. The identical subnetworks learn the embeddings or representations for a pair of inputs. These identical subnetworks are joined together by a distance metric to measure the differences in the representations for the pair of inputs. We use a radio signal as input to the two subnetworks of the Siamese network. The Siamese network outputs the distance metric between the representations computed from the given two inputs. In the case of packets from the same device, the representations for the inputs should be similar and hence the distance metric should ideally output 0, as shown in Figure 1a. Similarly, for packets from different devices, the representations should be different and the distance metric should ideally output d_{max}, the maximum distance in the representation space, as shown in Figure 1b.

We compute the similarity score for a pair of packets using Equation 1, where d is the distance metric computed by the Siamese network. To identify the source of an incoming packet, we compare it with the reference packets for all the devices using our Siamese network. We predict the device with the highest similarity score as the source for the incoming



(a) Operation for packets from the same device



(b) Operation for packets from different devices

Fig. 1: Operation of Siamese network for PHY-Layer identity verification

packet. We term this methodology as PLIO-Physical Layer Identification using One-shot learning.

$$Similarity\ Score = \frac{d_{max} - d}{d_{max}} \tag{1}$$

B. Overview of the datasets

To evaluate the performance of PLIO for different test scenarios, we create two datasets of time-domain radio signals. We store the time-domain radio signals as a sequence of In-Phase and Quadrature samples (I/Q samples) obtained from the SDRs. Frequency-domain signals emphasize the CFO for different devices. Although it is the most discriminatory feature between devices, it is susceptible to clock drift with temperature variations of the transmitter, as shown by Robyns et al. [22]. Hence, we use time-domain I/Q samples with our deep neural network.

The datasets contain unprocessed I/Q samples from commercial off-the-shelf (COTS) IEEE 802.15.4 IoT devices transmitting at 2.48GHz using Offset Quadrature Phase Shift Keying (OQPSK) modulation. The IoT devices are programmed using Contiki-NG to transmit the same data packet periodically. We collect I/Q samples for 1000 packets from each IoT node.

In this paper, we focus on the steady-state radio characteristics as these characteristics can be obtained using low-sampling rates on low-cost Software Defined Radios (SDRs). We collect the steady-state baseband radio samples with a Universal Software Radio Peripheral (USRP) B210 SDR at a sampling rate of 4MHz. We disable receiver impairments compensation and use unprocessed I/Q samples in our neural network. Receiver impairments compensation estimates the error introduced by the radio frequency components of the receiver and modifies the received signal. The receiver impairments compensation can remove important features from the radio signal. Furthermore, the correction factor introduced in different receivers can impact the received signal differently, making it hard to generalize the PHY-layer identification method to new receivers. Similar to receiver impairments compensation, channel equalization methods modify the received signal according to the predictions of a channel estimator. Grisenko et al. [11] have shown that channel equalization negatively impacts the performance of PHY-layer identification methods.

TABLE I: Dataset I

Platform	Chipset	Quantity	N	M
Zolertia Firefly	CC2538	10	7	3
Tmote Sky	CC2420	8	6	2
Zoleria Zoul	CC2538	10	7	3

TABLE II: Dataset II

Platform	Chipset	Quantity	N	M
Zolertia Firefly	CC2538	15	10	5
Thunderboard Sense2	EFR32MG12	6	4	2
nRF52840	nRF52840	5	3	2

We use the IEEE 802.15.4 physical layer implementation in GNU Radio by Bloessl et al. [6] to process the radio samples. We modify the physical layer to provide the sample index for the beginning and end of each radio packet. Using the Signal Metadata Format (SigMF) [1] format, we save the beginning and end of each packet together with the unprocessed radio samples. SigMF is a widely used format and opens the possibility of other works to test their methods on our dataset through standard libraries.

To evaluate the applicability of PLIO to different devices having different radio chipsets, we collect data from 54 devices across four radio chipsets. To evaluate the applicability of PLIO to new receivers and radio environments, we use three USRPs in two office environments to collect the datasets. Dataset 1 shown in Table I is collected using USRP1 in one office environment. Dataset 2 shown in Table II is collected using USRP2 and USRP3 in another office environment. The receiver is kept adjacent to the transmitter, and hence for this work, we assume the captured signal is not affected by multi-path effects and the signal power dominates the noise power.

C. Data Preprocessing

Next we describe our preprocessing methods that applied to the unprocessed I/Q samples.

Scaling: The absolute amplitude of a radio signal can be affected by the distance from the transmitter to the receiver and in-band radio noise. Scaling enables the deep neural network to learn the variations of the radio signal instead of the absolute amplitude of the radio signal, making it more generalizable to different radio conditions. We scale the I/Q samples corresponding to one packet using Equation 2.

$$X_{scaled} = A + \frac{(X - X_{min})(B - A)}{X_{max} - X_{min}} \tag{2}$$

X_{min} and X_{max} are the minimum and maximum value of unprocessed I/Q samples in a radio packet. A and B are the minimum and maximum values for the scaled I/Q samples. To ensure that the mean of the distribution of scaled I/Q samples in one packet is close to zero, we set A and B to -0.5 and 0.5 respectively. LeCun et al. [19] have shown that this improves the convergence of neural networks. As the distribution of the radio samples might contain unique characteristics of the

radio transmitter, we do not normalize the data distribution to a predefined mean.

Sliding Window: Different radio packets can have a variable number of I/Q samples. We use convolutional layers in our neural network that are restricted to operate on fixed-size inputs. Furthermore, variable size inputs cause the problem of vanishing gradients in deep neural networks. Hence, we use a sliding window over a radio packet to segment it into fixed-size segments of S I/Q samples, with the stride for the sliding window set to one I/Q sample. A sliding window also increases the robustness of the learning by providing shift-invariance of the features learned by the neural network [2]. We term these segments of S I/Q samples slices. We set S to 128 I/Q samples, which corresponds to 32 OQPSK symbols, each containing four I/Q samples.

D. Deep Neural Network Architecture for PLIO

We design the deep neural network architecture for feature extraction shown in Figure 2. Each subnetwork of our Siamese network uses this neural network architecture. Deep convolutional networks have been successfully used in image classification problems [18], [25]. We design our neural network by taking inspiration from these approaches and use a stack of three convolutional layers. We chose the number of convolutional layers empirically. We use one-dimensional (1D) convolutional layers to capture the local temporal features from the I/Q sample slices. Convolutional filters identify location-invariant features, which is essential as we work with fixed-length slices sampled from the radio packets. We append BatchNormalization layers to the convolutional layers to speed up the learning process by enabling predictive and stable behavior of the gradients [23]. Max Pooling layers reduce the dimensionality of the BatchNormalization layers' feature maps by selecting the most relevant features.

A layer of Long Short-Term Memory (LSTM) learns the sequential distribution of the features extracted by the convolutional layers. We output the internal state of the recurrent connections in the LSTM layer at each time step to learn the representations for each time step. A fully connected layer of 4096 neurons combines the representation from each time step of the LSTM layer to output the final feature vector in each subnetwork of the Siamese network. We use the Sigmoid activation function on the fully connected layer to normalize the output of each feature between [0,1].

As our twin subnetworks output high-dimensionality feature vectors, L1 distance is preferable to L2 distance as suggested by Agarwal et al. [4]. Hence, we use the L1 distance to measure the difference between the feature vectors extracted by the twin subnetworks. A single output neuron weighs the importance of each feature in the difference vector and maps it to the output through a sigmoid activation function.

E. Identification

Next we describe our two different identification methods, namely slice-level and packet-level identification.

1) Slice-level identification

We store a packet from each device in the IoT network as reference packet. To identify the source of a test slice, we first scale the slice using Equation 2. Next, we randomly sample a slice from the reference packet for each device and store them in a set that we term the support set. We generate a set of pairs of radio slices using Equation 3

$$Pair[i] = \{Test - Slice, Support - Set[i]\}, for\ 0 \leq i < N \tag{3}$$

where N is the number of devices currently in the IoT network. We provide the Siamese network with the pairs of slices. The model outputs the similarity score of the test slice for each of the reference slices in the support set. We identify the source of the reference slice with the highest similarity score as the transmitter.

2) Packet level identification

We perform the packet level identification in a one-shot learning setting with one reference packet for each device in the IoT network. We combine slice-level identification for an ensemble of slices to identify the source of a packet. We randomly sample K slices from the single reference packet for each of the N devices. For each of the K test slices, we perform slice-level identification. We combine the results across the K test slices using soft-voting [31] to consider the confidence of our neural network for each slice-level identification. We sum the similarity scores across the K test slices, and we identify the source of the reference packet with the highest sum as the transmitter.

IV. Implementation

In this work, we evaluate the generalization capability of our network to *unseen* devices. Hence, we first segment the two datasets into *seen* and *unseen* devices. We keep 30% of the devices for each platform as unseen devices. The number of *seen* and *unseen* devices used for the two datasets is shown as N and M respectively in Table I and Table II. Overall we train our neural network on 20 devices in Dataset-I, and 17 devices in Dataset-II. We randomly select 50 packets from each of the *seen* devices and process the corresponding I/Q samples using our data preprocessing methods to generate the radio slices. We split the processed slices into training set and validation set by randomly selecting 70% of the slices as our training slices. We use the remaining 30% of the slices as our validation-set that controls the training process.

A. Neural Network Hyperparameters

We implement our neural network using the Keras deep-learning framework in Python 3.6. We set the kernel size of the 1D convolutional filters to the length of a sequence of two OQPSK symbols for all the convolutional layers. This kernel size enables the convolutional filters to learn the phase transitions between two OQPSK symbols. Deeper convolutional layers combine the features extracted by the preceding convolutional layers. More filters in the deeper layers enable the neural network to learn more combinations

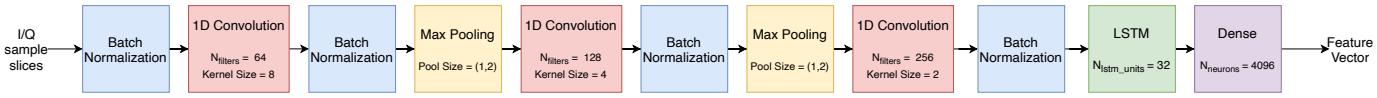


Fig. 2: Deep Neural Network Architecture

of features. Hence, we increase the number of filters for the three convolutional layers progressively from 64 to 128 to 256.

We use one Dropout layer after each of the convolutional layers and after the LSTM layer. Dropout randomly disables a fraction of the neural network connections and forces the network to learn from a partial representation of the previous layer. Hence, Dropout allows the neural network to generalize well to new inputs. We empirically set the Dropout rate to 0.2 for the convolutional layers and 0.4 for the LSTM layer. We term the neural network architecture with these hyperparameters as our model.

B. Training Procedure

We train our model with batches of pairs of input slices. We need examples of both similar and dissimilar pairs to avoid bias towards either similar or dissimilar pairs. To generate a similar pair, we randomly select a device from our training set and sample two random slices. To generate a dissimilar pair, we randomly select two devices from our training set and sample a random slice from each of them. We present 50% similar pairs and 50% dissimilar pairs to the model in each batch. We assign the target output for each similar pair to one, and for dissimilar pairs, we set the target output to zero.

To make our model robust to noisy radio environments, we need to train the model with packets for different noisy radio environments. However, collecting packets for various controlled radio environments can be cumbersome. Data augmentation through the addition of training noise allows us to create packets for different noisy radio environments from a single packet [30]. Hence, we randomly add a mixture of Additive White Gaussian Noise (AWGN) signals of different power levels to the training slices. We specify the maximum noise power for the mixture as P dBm during the training process. We randomly select a noise power from the set, 0, 1, 2, ... P dBm, and generate an AWGN signal with a mean of zero and standard deviation equal to the square root of the noise power in volts and combine it to a random slice. We iterate this process for 35% of the training slices.

As the goal of our model is to learn a similarity metric for a given pair of inputs, we use the contrastive loss introduced by Chopra et al. [8] to train our model. Contrastive loss minimizes the distance between representations in each subnetwork of our model for similar pairs and maximizes the distance between representations for dissimilar pairs. Given a model parametrized by θ, the contrastive loss function is defined by Equation 4.

$$L(\theta) = \frac{1}{2}(1-Y)(\hat{Y}^2)) + \frac{1}{2}Y\{max(0, d_{max} - \hat{Y})\}^2 \quad (4)$$

In this equation, Y is the target output for the model. The L1 distance between the representations of each subnetwork in the model for a pair of inputs is $\hat{Y}$. d_{max} is the maximum L1 distance for dissimilar pairs. In our model, $d_{max} = 1$, since we map the representations of each subnetwork of our neural network to [0,1] using the Sigmoid activation function.

We train the model for 100 iterations. In each training iteration, we sample 5000 batches of 512 pairs of slices from the training set to train the model. After each iteration, we sample 100 batches of 512 pairs of slices from the validation set to validate the model. Each iteration, we measure the averaged contrastive loss for the validation pairs as our validation loss. We save the model if the validation loss is lower than the validation loss from the previous iteration. We halt the training process if the validation loss does not improve over ten iterations. We use the model with the lowest validation loss for our evaluations to prevent overfitting to the training set.

V. Evaluation

We follow a quantitative experimental methodology to evaluate PLIO.

A. Metrics

We investigate the performance of PLIO using the following metrics:

- **Precision:** Precision measures the proportion of accurate identifications for each device. With high precision for identification, a lower number of packets from other devices get wrongly identified as the correct devices. Particularly in PHY-layer identification, we desire high precision as we do not want to grant rogue devices access to an IoT network.
- **Recall:** Recall measures what proportion of the total number of the radio packets for a particular device we identified correctly. High recall enables us to identify packets from a device correctly, preventing unnecessary retransmissions, thus preserving energy for the IoT devices.

B. Testing Process

We term the devices used in an experiment as our test devices. To perform packet-level identification, we create two sets of packets: test-set and support-set. The support-set consists of reference packets for each test device. To ensure that PLIO is robust to the choice of reference packet, we randomly select 50 packets from each test device as our support set. For our test-set, we randomly select 300 packets for each test device. We iterate this process five times to evaluate PLIO with multiple combinations of reference packets and test packets. As in our training process, we add AWGN signals to the packets in the test-set to measure the impact of noise on our identification performance. For each packet in the test-set, we perform packet-level identification using the method described in Section III-E2.

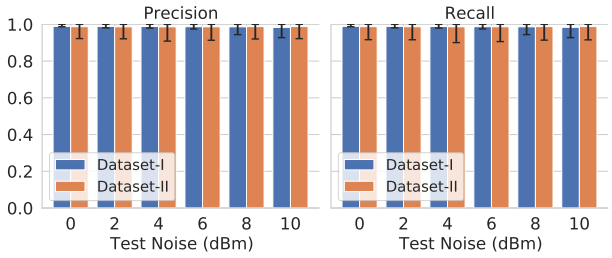


Fig. 3: Precision and recall for identification of the *seen* devices in Dataset-I and Dataset-II versus test noise power (P = 10 dBm, K = 25).

C. Evaluation Setup

We evaluate PLIO with respect to different training noise, the number of slices used for packet-level identification, and devices used for training the model. We define the maximum noise power used for training the model as P. We define the ensemble size (number of slices) used for packet-level identification as K. The devices used in training the neural network define the training scenario. We use the training scenarios shown in Table III in our experiments.

TABLE III: Training scenarios

Scenario	Description
T1	Trained on the *seen* devices in Dataset-I
T2	Trained on the *seen* devices in Dataset-II
C	Trained on the *seen* devices in both Dataset-I and Dataset-II

We perform our evaluations on a virtual machine equipped with 32 processor cores, 128 GB of RAM, and a Nvidia 2080 Ti GPU. Following our problem formulation, we first verify that PLIO can identify the *seen* devices in Dataset-I and Dataset-II. Next, we evaluate if PLIO can generalize to the *unseen* devices in Dataset-I and Dataset-II. Finally, we evaluate PLIO for identification of both *seen* and *unseen* devices.

D. Seen Device Identification

As a baseline for our model, we investigate if PLIO can identify the *seen* devices in noisy radio environments. Figure 3 shows the precision and recall for identification of *seen* devices in Dataset-I and Dataset-II for training scenario *C*. The bar plot shows the mean of the precision and recall for the identification of packets from all the devices. We plot the 95 percentile and five percentile values of the distribution of precision and recall across all devices over five iterations with the error bars. The X-axis shows the power of the noise signal, which we add to each test packet. Figure 3 shows that PLIO can identify new packets from the *seen* devices with high precision and recall even in noisy radio environments. On average, PLIO takes 70 ms to compare a test packet with the reference packet for ensemble size (K) of 25.

E. Unseen Device Classification

In this section, we evaluate the performance of PLIO for identifying the *unseen* devices in Dataset-I and Dataset-II for different evaluation parameters and training scenarios.

1) Adaptability

Traditional classifiers must be trained for specific IoT networks and their devices which prevents the adaptability to different IoT networks. As our neural network model provides a similarity score for a pair of input slices, it is sufficient to train a single model that works well in all IoT networks. We compare the performance of a model trained using scenario *C* to the performance of models trained using scenario *T1* and *T2*. *T1* and *T2* represent models trained for a specific IoT network, whereas *C* represents a single model for multiple networks.

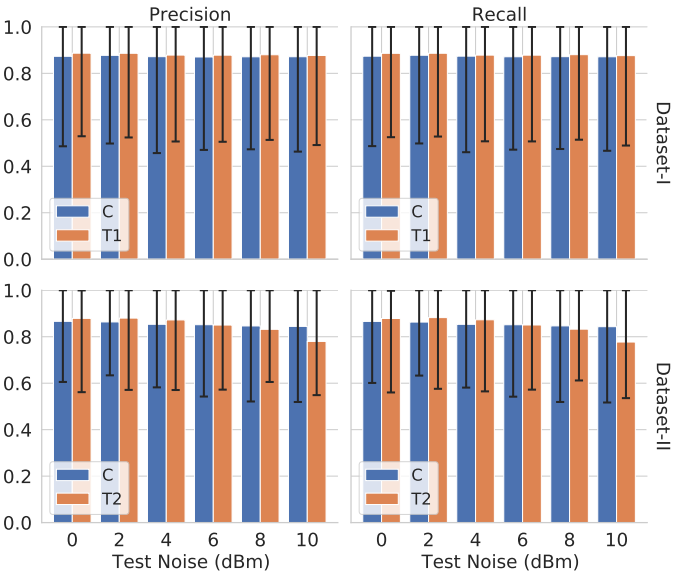


Fig. 4: Comparison of precision and recall for identification of the *unseen* devices in Dataset-I and Dataset-II for different training scenarios versus test noise power (P = 10 dBm, K = 25).

Figure 4 shows the comparison of precision and recall for identification of *unseen* devices in Dataset-I and Dataset-II using our training scenarios *T1*, *T2*, and *C*. Scenario *C* achieves similar results to those obtained using scenario *T1* for Dataset-I. For Dataset-I, across all test noise powers, PLIO on average identifies the *unseen* devices with a precision and recall above 85%. Similarly, PLIO achieves a precision and recall for identification above 80% for all test noise powers using scenario *C* for Dataset-II. In the case of Dataset-II, while scenario *T2* is better than scenario *C* at low noise environments, scenario *C* adapts well to high noise environments and outperforms scenario *T2*. Scenario *C* is trained with more diverse data consisting of devices from both Dataset-I and Dataset-II, which makes it more robust to variations of the I/Q samples with noise.

Two Thunderboard Sense2 devices in Dataset-II and two Firefly devices in Dataset-I get misidentified with each other. As shown in Section V-E5, PLIO does not adapt well to these two chipsets. Hence, PLIO has large variance in the identification performance of individual devices as shown in Figure 4.

2) Impact of training noise

We investigate if the addition of noise to the training data improves the identification performance of PLIO in noisy radio environments.

Figure 5 shows the comparison of precision and recall for the identification of *unseen* devices for different training noise

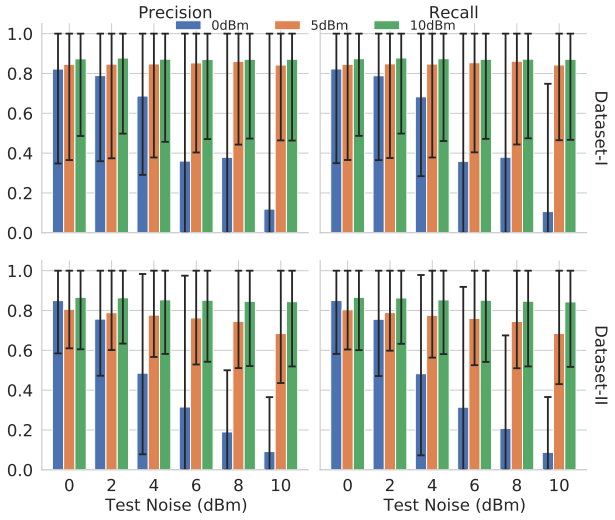


Fig. 5: Precision and recall for identification of *unseen* devices in Dataset-I and Dataset-II for different training noise power (P) versus test noise (training scenario = C and K = 25). Data augmentation with AWGN signals, particularly AWGN signals with high power, during training makes PLIO robust to variations of I/Q samples in noisy test environments.

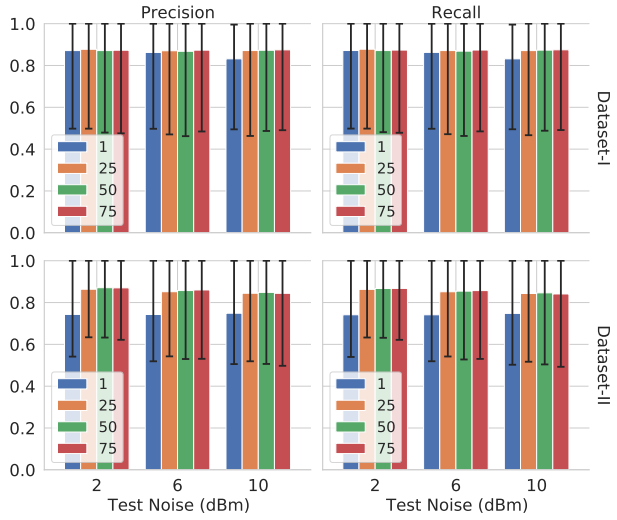


Fig. 6: Precision and recall for identification of *unseen* devices in Dataset-I and Dataset-II for different ensemble sizes (K) versus test noise (training scenario = C and P = 10 dBm). Packet-level identification improves the precision and recall of PLIO.

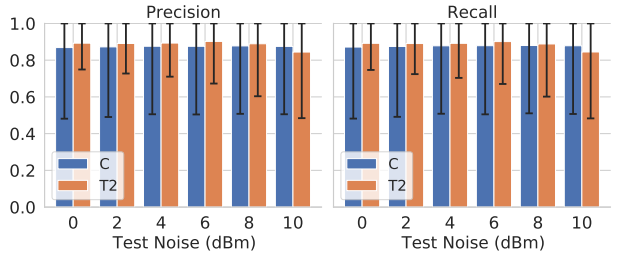


Fig. 7: Comparison of precision and recall for identification of *unseen* devices using different receiver for different training scenarios versus test noise (P = 10 dBm, K = 25). PLIO generalizes well to *unseen* devices with a different receiver, with high precision and recall even in the presence of 10 dBm test noise.

powers at different test noise powers for Dataset-I and Dataset-II. The figure shows that augmenting the training data with AWGN signals of high power improves both the mean precision and recall for our model for both datasets. Furthermore, such data augmentation improves the adaptability of PLIO to noisy test scenarios. PLIO achieves mean precision and recall of over 80% for *unseen* devices of both datasets even with the addition of 10 dBm test noise with data augmentation. There is a considerable degradation in the identification performance of PLIO for noisy test environments when we do not augment the data with training noise. The addition of training noise also reduces the variance of precision and recall for identification among test devices.

3) Impact of ensemble size

PLIO is dependent on the ensemble size K used for packet-level identification of each packet. We can reduce the inference time for the identification and the storage needed for the reference packets by reducing the ensemble size. However for small ensemble sizes, the identification is performed on a small subset of a packet's slices, which might be inaccurate. Hence, we investigate the performance of packet-level identification for different ensemble sizes.

Figure 6 shows the precision and recall for identification of *unseen* devices of Dataset-I and Dataset-II for different ensemble sizes. The top-row of Figure 6 shows similar precision and recall results across all test noise powers for different ensemble sizes for Dataset-I. However, in the case of *unseen* devices in Dataset-II, using an ensemble of slices yields better precision and recall than identification with one slice, as shown by the bottom row of Figure 6.

In the case of *unseen* devices in Dataset-II, some slices of Firefly and Thunderboard Sense2 devices get misidentified. We hypothesize this is because these devices have similar radio characteristics. Packet-level identification using an ensemble considers the results and confidence of the model across several slices, making the model robust to slice-level differences in a packet. Hence, packet-level identification using an ensemble improves the precision and recall for identification. Furthermore, we can conclude that increasing the ensemble size over 25 does not further improve the identification performance.

4) Generalization to unseen receiver

To evaluate the impact of *unseen* receiver on PLIO, we investigate the identification performance of PLIO for *unseen* devices when using a different receiver.

We collect I/Q samples for 1000 packets from the *unseen* devices of Dataset-II using another USRP. In total, we have nine *unseen* devices in this dataset across three different radio chipsets. We perform the evaluation using 300 packets from each of the *unseen* devices.

Figure 7 shows the comparison of precision and recall for the training scenarios C and $T2$. Figure 7 shows that PLIO

generalizes well to a different receiver. PLIO identifies *unseen* devices with a different receiver across all test noise powers with precision and recall above 80%. As shown in Figure 7, scenario *C* outperforms scenario *T2* in noisy test environments. Scenario *C* is trained across multiple receivers with a much higher diversity of training devices. Hence, training with scenario *C* is more robust against receiver-specific features.

5) Generalization to unseen platforms

The features used to discriminate between devices of different platforms might be different. This would limit the adaptability of PLIO to *unseen* devices with an *unseen* platform. Hence, we investigate how PLIO performs for *unseen* devices with an *unseen* platform.

We select one of the five platforms in our datasets as our test platform. We train our model on 50 packets from all devices except the devices from the test platform using our aforementioned training process. We perform the evaluation using 300 packets from each of the devices with the test platform. We iterate this process for all the platforms in our datasets.

Figure 8 shows the precision and recall for identification of *unseen* devices with different *unseen* platforms for different test noise powers. There is a significant variation in the performance of our model for different test platforms.

Although the Firefly and Zolertia Zoul platforms share the same radio chipset, PLIO only achieves a precision and recall of around 45% for the Firefly devices (random chance for Firefly devices is 4%). The precision and recall for Firefly devices are influenced by the number of Firefly devices, as shown by the high variance in Figure 8. We hypothesize that the large device diversity for the Firefly devices enables the model to learn features that apply to Zolertia Zoul devices. The small device diversity for the Zolertia Zoul devices limits the adaptability of the features learned by the model to the identification of Firefly devices. Furthermore, we hypothesize that the 2.4GHz transmitter in the CC2538 and CC2420 has similar characteristics, which explains why our model adapts well to the Tmote Sky platform.

PLIO does not adapt well to the nRF52840 and Sense2 devices. These two platforms have multi-protocol radio chipsets that are significantly different from the other radio chipsets in our datasets. This limits the adaptability of PLIO to *unseen* devices with these platforms.

Our results show that, PLIO can only be applied to *unseen* devices with a radio chipset similar to the devices in the training set. Furthermore, PLIO improves with the diversity of devices for a chipset.

F. Extending an IoT network

The *unseen* devices need be to distinguished from both other *unseen* devices as well as *seen* devices. Because of the asymmetrical nature of information available to the model about the *seen* and *unseen* devices, *unseen* devices might have high chances of being identified as one of the *seen* devices. Furthermore, minimal retraining of the model can improve the identification performance of the model. Hence, we investigate

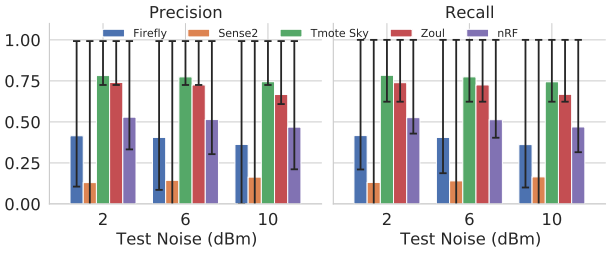


Fig. 8: Comparison of precision and recall for identification of *unseen* devices with *unseen* platform versus test noise (P = 10 dBm, K = 25). The radio chipset in a platform has a significant impact on the adaptability of PLIO to *unseen* devices with *unseen* platforms.

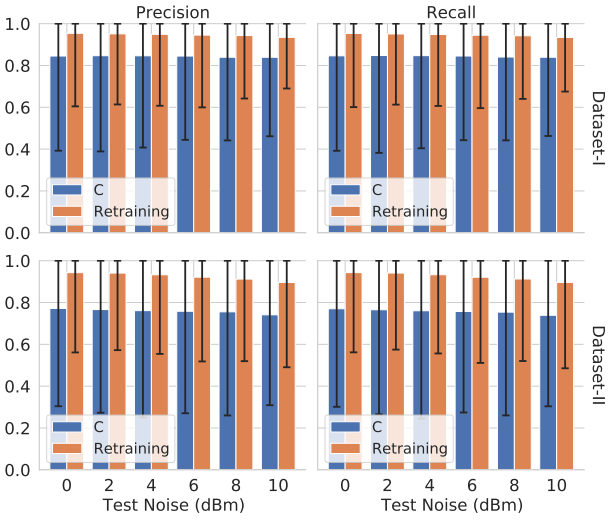


Fig. 9: Precision and recall for identification of all devices in Dataset-I and Dataset-II for training scenario *C* and retraining scenario versus test noise. (K = 25). Minimal retraining can significantly improve the identification performance of PLIO.

how PLIO performs in such scenarios using Dataset-I and Dataset-II with and without retraining.

To compare how minimal training affects the performance of our model, we retrain our model with three radio packets from all devices in Dataset-I and Dataset-II for five training iterations, using the training process described previously. The minimal training process takes approximately 20 minutes on our evaluation setup. We set the maximum power (P) for the mixture of AWGN signals added to the training data to 10 dBm, as our previous experiments have shown that this results in the highest precision and recall.

Figure 9 shows the results for Dataset-I and Dataset-II for two scenarios: training scenario *C* and retraining scenario against test noise. With training scenario *C* for both the datasets, PLIO adapts well to test noise with similar precision and recall across all test noise powers. However, PLIO achieves a higher mean (86%) and lower variance for identification of devices in Dataset-I, in comparison to the mean (76%) and variance for identification of devices in Dataset-II. As shown in Section V-E5, there is a high variance in the performance of PLIO for *unseen* devices with different

radio chipsets. As Dataset-I and Dataset-II consist of different radio chipsets, their results differ.

Figure 9 shows that the minimal retraining of the network improves the performance of PLIO for both Dataset-I and Dataset-II, achieving a precision and recall above 93%. Minimal retraining also decreases the variance of identification performance of individual devices. However, while PLIO adapts well to different test noise power for the devices in Dataset-I, the precision and recall for the devices in Dataset-II decreases slightly, and the variance increases at higher test noise powers. The generalization capability of PLIO with minimal retraining is limited, and hence the discriminating features learned do not adapt well to increasing noise levels when two devices have similar characteristics. Nevertheless, with minimal retraining, PLIO achieves 89% precision and recall across all devices in Dataset-II in the presence 10 dBm test noise.

VI. Conclusion

This paper has presented PLIO a one-shot learning based PHY-layer identification method, for identification of both *seen* and *unseen* devices. We have shown that PLIO identifies *unseen* devices using a reference set of packets with precision and recall over 80% in the presence of 10 dBm noise in the test signals. Furthermore, we demonstrate the adaptability of PLIO to different IoT networks even when a different receiver is used. Finally, we showcase that minimal retraining of PLIO using only three packets from each device we can identify all devices in the datasets with a precision and recall of 93%.

VII. Acknowledgements

This work has been (partially) funded by the H2020 EU/TW joint action 5G-DIVE (Grant #859881), and VINNOVA.

References

[1] The Signal Metadata Format (SigMF), 0.0.2, Jan 2019. [Online; accessed 2021-04-29].

[2] A. Al-Shawabka et al. Exposing the fingerprint: Dissecting the impact of the wireless channel on radio fingerprinting. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 646–655. IEEE, 2020.

[3] I. Agadakos, N. Agadakos, J. Polakis, and M. R. Amer. Deep complex networks for protocol-agnostic radio frequency device fingerprinting in the wild. *CoRR*, abs/1909.08703, 2019.

[4] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer, 2001.

[5] S. Aheleroff, X. Xu, Y. Lu, M. Aristizabal, J. Pablo Velásquez, B. Joa, and Y. Valencia. Iot-enabled smart appliances under industry 4.0: A case study. *Advanced Engineering Informatics*, page 101043, 2020.

[6] B. Bloessl, C. Leitner, F. Dressler, and C. Sommer. A GNU radio-based ieee 802.15. 4 testbed. *12. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN 2013)*, pages 37–40, 2013.

[7] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a" siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, pages 737–744, 1993.

[8] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 539–546, 2005.

[9] I. Cvitić, D. Peraković, M. Periša, and B. Gupta. Ensemble machine learning approach for classification of iot devices in smart home. *International Journal of Machine Learning and Cybernetics*, pages 1–24, 2021.

[10] A. Gadre, R. Narayanan, A. Luong, A. Rowe, B. Iannucci, and S. Kumar. Frequency configuration for low-power wide-area networks in a heartbeat. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 339–352, 2020.

[11] A. Gritsenko, Z. Wang, T. Jian, J. Dy, K. Chowdhury, and S. Ioannidis. Finding a 'new'needle in the haystack: Unseen radio detection in large populations using deep learning. In *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 1–10, 2019.

[12] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pages 1735–1742, 2006.

[13] T. hoon Kim, C. Ramos, and S. Mohammed. Smart city and IoT. *Future Generation Computer Systems*, pages 159–162, 2017.

[14] K. Sankhe et al. ORACLE: Optimized Radio clAssification through Convolutional neuraL nEtworks. *IEEE INFOCOM*, 2019-April:370–378, 2019.

[15] K. Sankhe et al. No Radio Left Behind: Radio Fingerprinting Through Deep Learning of Physical-Layer Hardware Impairments. *IEEE Transactions on Cognitive Communications and Networking*, 6(1):165–178, 2020.

[16] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[17] J. Kotak and Y. Elovici. Iot device identification using deep learning. In *Computational Intelligence in Security for Information Systems Conference*, pages 76–86. Springer, 2019.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012.

[19] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[20] W. Pei, D. Tax, and L. V. D. Maaten. Modeling time series similarity with siamese recurrent networks. *ArXiv*, abs/1603.04713, 2016.

[21] P. K. D. Pramanik, B. K. Upadhyaya, S. Pal, and T. Pal. Internet of things, smart sensors, and pervasive systems: Enabling connected and pervasive healthcare. In *Healthcare Data Analytics and Management*, pages 1–58. 2019.

[22] P. Robyns, E. Marin, W. Lamotte, P. Quax, D. Singelée, and B. Preneel. Physical-layer fingerprinting of LoRa devices using supervised and zero-shot learning. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 58–63, 2017.

[23] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How does batch normalization help optimization? *arXiv preprint arXiv:1805.11604*, 2018.

[24] Y. Shi, K. Davaslioglu, Y. E. Sagduyu, W. C. Headley, M. Fowler, and G. Green. Deep learning for RF signal classification in unknown and dynamic spectrum environments. In *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 1–10, 2019.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[26] T. Jian et al. Deep learning for rf fingerprinting: A massive experimental study. *IEEE Internet of Things Magazine*, 3(1):50–57, 2020.

[27] L. J. Wong, W. C. Headley, S. Andrews, R. M. Gerdes, and A. J. Michaels. Clustering Learned CNN Features from Raw I/Q Data for Emitter Identification. *Proceedings - IEEE Military Communications Conference MILCOM*, pages 26–33, 2019.

[28] N. Yousefnezhad, A. Malhi, and K. Främling. Automated iot device identification based on full packet information using real-time network traffic. *Sensors*, 21(8):2660, 2021.

[29] J. Yu, A. Hu, G. Li, and L. Peng. A Robust RF Fingerprinting Approach Using Multisampling Convolutional Neural Network. *IEEE Internet of Things Journal*, pages 6786–6799, 2019.

[30] X. Zhou, A. Hu, G. Li, L. Peng, Y. Xing, and J. Yu. Design of a Robust RF Fingerprint Generation and Classification Scheme for Practical Device Identification. *2019 IEEE Conference on Communications and Network Security, CNS 2019*, 6(1):196–204, 2019.

[31] Z. Zhou and X.-Y. Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Trans. Knowl. Data Eng.*, 18:63–77, 2006.