ON THE EXTRACTION OF RF FINGERPRINTS FROM LSTM HIDDEN-STATE VALUES FOR ROBUST OPEN-SET DETECTION

Luke Puppo[1], Weng-Keen Wong[1], Bechir Hamdaoui[1], Abdurrahman Elmaghbub[1], Lucy Lin[1]
[1]School of EECS, Oregon State University, Corvallis, OR USA 97331

NOTE: Corresponding author: Weng-Keen Wong, wongwe@oregonstate.edu

Abstract – *New capabilities in wireless network security have been enabled by deep learning that leverages and exploits signal patterns and characteristics in Radio Frequency (RF) data captured by radio receivers to identify and authenticate radio transmitters. Open-set detection is an area of deep learning that aims to identify RF data samples captured from new devices during deployment (aka inference) that were not part of the training set; i.e. devices that were unseen during training. Past work in open-set detection has mostly been applied to independent and identically distributed data such as images. In contrast, RF signal data present a unique set of challenges as the data forms a time series with non-linear time dependencies among the samples. In this paper, we introduce a novel open-set detection approach for RF data-driven device identification that extracts its neural network features from patterns of the hidden state values within a Convolutional Neural Network Long Short-Term Memory (CNN+LSTM) model. Experimental results obtained using real datasets collected from 15 IoT devices, each enabled with LoRa, wireless-Wi-Fi, and wired-Wi-Fi communication protocols, show that our new approach greatly improves the area under the precision-recall curve, and hence, can be used successfully to monitor and control unauthorized network access of wireless devices.*

Keywords – Deep learning, network authentication, open-set detection, radio frequency

1. INTRODUCTION

The proliferation of Internet Of Things (IoT) devices in sensitive environments, such as military bases, government buildings, and private businesses, creates a need for detecting anomalous devices that pose security threats. These devices can easily bypass security measures as they can be concealed. Traditional detection methods are ineffective at identifying unauthorized wireless devices, especially with attacks like cloning and man-in-the-middle [1].

1.1 Deep learning-based RF Fingerprinting

RF fingerprinting is a recognized key method to enhance security in IoT networks [2, 3, 4]. It extracts device-specific features from RF signals to identify wireless transmitters, leveraging unique hardware imperfections during transmitter manufacturing. Feature extraction methods range from hand-crafted to deep learning-based approaches that identify features from raw RF signals. This paper proposes `HiNoVa`, a new machine learning-based open-set detection method that identifies unauthorized (also referred to as unknown or unseen) IoT devices and authorized (also referred to as known or seen) devices. `HiNoVa` is tested on datasets collected from devices using LoRa and Wi-Fi protocols [5]. LoRa is a wireless communication technology designed for IoT devices that operates in the sub-gigahertz frequency range, enabling long-range, low-power, bidirectional communication. LoRa's advantages include longer range, better penetration through obstacles, and low power consumption, making it suitable for IoT applications that require a wide area network coverage. However, LoRa has lower data rates than Wi-Fi, making it unsuitable for high-speed data transfer applications. The crowded sub-gigahertz frequency range can also lead to interference from other devices. Each of the protocols, LoRa and Wi-Fi, has its practical use and is commonly adopted by various transmitters, and hence, our proposed open-set detection method is tested using both protocols.

1.2 Open-set detection

Supervised machine learning algorithms are increasingly being used to perform RF fingerprinting [6]. However, these algorithms typically operate under *closed-set* recognition, meaning that they assume the classes encountered during testing are identical to those seen during training. This means that if a Neural Network (NN) is trained to identify the two classes of cats and dogs, it fails to recognize an unknown type of animal, such as a bird, as a distinct animal and will instead misclassify it as either a cat or a dog. This limitation is particularly problematic in real-world scenarios where device fingerprinting is used for security purposes such as network access authentication. In this authentication use case, the classes correspond to *known or legitimate* devices and it is crucial for the system to accurately detect *unknown or illegitimate* devices (i.e. the open-set devices) to raise security alerts.

For these types of problems, *open-set* detection [7] can be used, where the classifier needs to recognize that data samples do not belong to any of the known devices seen during training, and raises an alert when this happens. Our work introduces HiNoVa, a novel open-set detection approach for authenticating wireless devices using RF fingerprinting.

1.3 Contributions

We present HiNoVa, a novel open-set detection method for wireless communication protocols. HiNoVa leverages the Hidden Node Values within a trained Long-Short-Term Memory (LSTM) unit of a deep NN to generate a unique device fingerprint for each known device. Then, new fingerprints encountered during deployment can be compared against the fingerprints of known devices, enabling the system to accurately identify unknown devices. After undergoing training on a set of known devices, the open-set detection process is highly efficient and can be performed in real time even on consumer-grade devices. This makes HiNoVa an ideal solution for wireless security applications, where the ability to quickly identify unauthorized/unknown devices is of utmost importance.

The paper is structured as follows: Section 3 presents the machine learning architecture used by our method. Section 4 presents the details of the HiNoVa algorithm. Section 5 describes the LoRa, wireless-Wi-Fi, and wired-Wi-Fi datasets used in our evaluation and Section 6 evaluates the performance of HiNoVa using these datasets. The last section concludes the paper. For ease of reference, Table 1 contains a list of acronyms used in this publication, along with their meanings.

Table 1 – Acronyms that are used in this publication, along with their meanings.

Acronym	Meaning
AUPRC	Area Under the Precision-Recall Curve
CNN	Convolutional Neural Network
CSS	Chirp Spread Spectrum
DTW	Dynamic Time Warping
I.I.D.	Independently and Identically Distributed
IoT	Internet of Things
IQ	In-phase and Quadrature
LSTM	Long Short-Term Memory
NN	Neural Network
ReLU	Rectified Linear Unit
RF	Radio Frequency
RNN	Recurrent Neural Network
USRP	Universal Software Radio Peripheral

2. RELATED WORK

Open-set detection is an area of machine learning that has attracted considerable attention and the literature is vast. We will refer the reader to a survey such as [8] and in this section, we will only highlight the most closely related approaches.

One of the simplest approaches to open-set detection is to use the predicted class probability as an indicator of the model's confidence that the data instance belongs to one of the known devices [9]. In an NN, the predicted class probability is the maximum class probability output by a softmax distribution. If this value is low, it indicates that the instance is likely from an unknown device.

Recent work [10, 11] shows that the maximum logit score (which we refer to as *MaxLogit*) is a stronger baseline for detecting open-set instances. Logits are the outputs of the last linear layer of a deep neural network. In classification, these logits are the inputs to the softmax layer, which normalizes the logits to be a valid probability. Normalizing the logits removes information about their raw magnitude, which is valuable for detecting open-set instances [10]. The MaxLogit score is the value of the largest logit, which is indicative of the uncertainty of the classifier as to the device; an open-set instance should have a lower maximum logit value.

Recent approaches to open-set detection focus on leveraging internal node values and activation patterns of neurons inside neural networks to detect open-set samples. For example, ReAct [12] analyzes the internal activations of neural networks and identifies highly distinctive signature patterns for open-set distributions. Dietterich et al. [11] argue that detecting novel objects in object recognition applications with an open set of possible categories is a familiarity-based problem rather than a novelty-based problem. Their familiarity hypothesis posits that state-of-the-art methods based on the computed logits of visual object classifiers succeed by detecting the absence of familiar learned features rather than the presence of novelty.

Much of the literature for open-set detection applies to data instances that are independent and identically distributed (i.i.d). To our knowledge the only work for open-set detection on time series is by Akar et al. [13], which clusters the time series in each known class to identify a class-specific barycenter. Then, during deployment, new time series are identified by how close they are to these barycenters, where the closeness is determined by Dynamic Time Warping (DTW) and also by cross-correlation. Time series that are not close to the barycenters of known devices are flagged as an unknown device. DTW has a complexity of $O(T^2)$, where T is the length of the two time series to be aligned. The algorithm by Akar et al. uses DTW in the inner loop of several operations and is extremely computationally expensive.

A handful of papers have applied open-set detection to RF fingerprinting. Gritsenko et al. [14] use the maxi-

mum probability from the softmax layer and the ratio of slices predicted to belong to each device to establish the confidence in the device prediction. Hanna et al. [15] investigate a variety of methods such as the maximum softmax probability and methods that incorporate data from known unauthorized devices. Gaskin et al. [16, 17] propose Tweak, a lightweight calibration approach that leverages metric learning to achieve high open-set accuracy without the need for model retraining, making it more suitable for resource-constrained applications. In a recent work, Karunaratne et al. [18] use generative deep learning models to produce synthetic data from unauthorized devices, which are used to augment the training set. Our approach differs from these approaches by modeling the sequential nature of the time series data with a CNN+LSTM and leveraging these sequential relationships for open-set detection.

Another closely related area to open-set detection is anomaly detection [19]. In anomaly detection, the goal is to identify individual outliers that are rare with respect to the nominal (i.e. "normal") data instances. Anomaly detection has some subtle differences with open-set detection. First, in open-set detection, data instances from the unknown class come from a semantically coherent grouping that is different from the known classes. In contrast, the anomalies found by anomaly detection need not form a coherent grouping. Second, the anomalies in a typical anomaly detection setting make up a small fraction of the data, with the nominal instances forming a large proportion of the data. In open-set detection, the unknown classes can potentially contain a large number of data instances. Despite these subtleties, anomaly detection techniques can, in some cases, be applied to open-set detection and vice versa; however, open-set detection methods have been found to outperform anomaly detection methods for detecting unknown devices [20].

3. THE CNN-LSTM NEURAL NETWORK ARCHITECTURE

In deep learning, a Recurrent Neural Network (RNN) layer is a layer type that allows for the processing of sequential data such as a time series by maintaining a memory state that can store information about the recent past. It consists of a single time step of the RNN, which involves computing a hidden state vector h_t and an output vector y_t at each time step t. The vector h_t depends not only on the input vector x_t at time step t, but also on the hidden state vector h_{t-1} at the previous time step. This dependence allows the network to maintain a memory of past inputs and use this information to inform its current output.

One limitation of this RNN layer is that it can have difficulty remembering long-term dependencies in the input sequence. To overcome this difficulty, the Long Short-Term Memory (LSTM) [21] layer was developed to handle long-term dependencies in the input sequence more effectively.

3.1 Long-Short-Term Memory (LSTM) layer

The LSTM layer consists of the following equations, where $\odot$ represents an element-wise product:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \quad (1)$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \quad (2)$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \quad (3)$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \quad (4)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (5)$$
$$h_t = o_t \odot \tanh(c_t) \quad (6)$$

Each term in the LSTM equations is described below:

- x_t: The input vector at time t.
- h_{t-1}: The previous hidden state vector.
- i_t, f_t, g_t, o_t: The input gate, forget gate, cell gate, and output gate activation vectors, respectively.
- c_t: The memory cell content vector, containing old memory cell content and newly added cell content.
- W_{ii}, W_{if}, W_{ig}, W_{io}: The weight matrices for input gates, forget gates, cell gates, and output gates for the input vector.
- W_{hi}, W_{hf}, W_{hg}, W_{ho}: The weight matrices for the input gates, forget gates, cell gates, and output gates for the previous hidden state.
- b_{ii}, b_{if}, b_{ig}, b_{io}: The bias vectors for the input gates, forget gates, cell gates, and output gates for the input vector.
- b_{hi}, b_{hf}, b_{hg}, b_{ho}: The bias vectors for the input gates, forget gates, cell gates, and output gates for the previous hidden state.
- h_t: The hidden state at time t.

The LSTM network has a cell state that can store information for long periods of time, and three gates that control the flow of information: the input gate, forget gate, and output gate. The input gate controls the input to the cell state, the forget gate controls how much of the previous cell state is retained, and the output gate controls the output from the cell state.

At each time step, the LSTM network takes an input x_t, the previous hidden state h_{t-1} and the previous cell state c_{t-1}, and uses these to compute the input gate i_t, forget gate f_t, cell gate g_t, and output gate o_t.

The cell state c_t (Eqn *(5)*) is updated based on the input gate i_t, forget gate f_t, and cell gate g_t. The input gate (Eqn *(1)*) controls how much new information is added to the

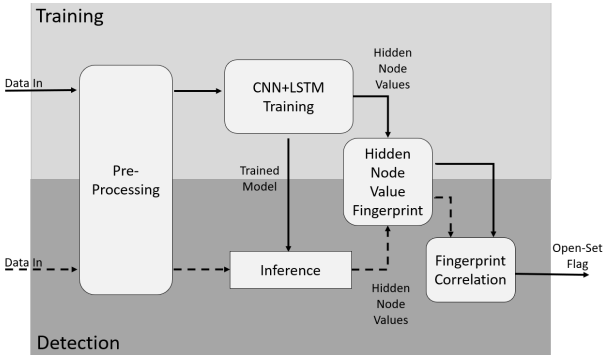


Fig. 1 – An overview of the HiNoVa framework.

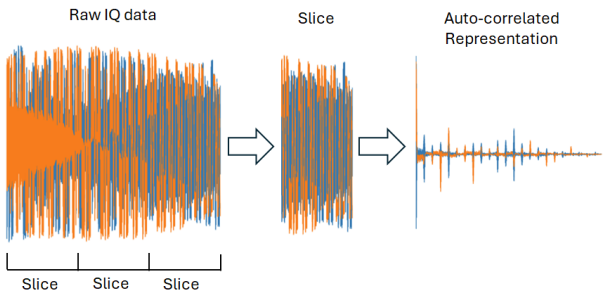


Fig. 2 – An overview of the preprocessing pipeline.

cell state and the forget gate (Eqn *(2)*) controls how much old information is retained. The cell gate (Eqn *(3)*) controls what new information is added to the cell state, by applying an activation function (e.g. $tanh$) to the input and previous hidden state.

Finally, the output gate o_t (Eqn *(4)*) controls how much of the current cell state is output as the new hidden state h_t. The new hidden state (Eqn *(6)*)is computed by applying the $tanh$ function to the updated cell state c_t and then multiplying it by the output gate o_t. The hidden state now contains both short and long-term memory, making it an excellent choice for an informative latent description.

3.2 Convolutional Neural Network LSTMs

Convolutional Neural Networks (CNNs) have been successful at image recognition because of their locality bias, which assumes that nearby pixels are useful in identifying an object. The key component of a CNN responsible for this locality bias is the convolutional layer, which convolves a set of filters to the input data in order to extract local features. The filters are typically small in size and slide over the input data in a sequential, linear fashion. This results in a feature map that highlights patterns in the input data and these patterns have the property of translational invariance (i.e. moving a cat a few pixels over still makes the cat present in the image).

A CNN can also be combined with an LSTM layer by piping the output of the convolutional layer into the LSTM. We call this hybrid a CNN+LSTM, which is well suited for discovering patterns in RF transmissions, which have cyclic patterns over time that are predictive of the device.

4. METHODOLOGY

Fig. 1 provides an overview of the entire HiNoVa algorithm and illustrates how each component interacts with the others. The top half shows how the training data is processed and the bottom half represents the detection phase operating on test data.

4.1 Preprocessing

The data captured from IoT devices during testing is initially processed and stored in the In-phase and Quadrature (IQ) format. The IQ components of an RF signal are crucial in accurately reproducing the original signal and are represented as complex numbers, with the real and imaginary values represented by I and Q, respectively. During testing, each IoT device sends a 20-second message, which is captured by a USRP receiver and saved in a complex number format.

To preprocess the data for analysis, the complex numbers are converted back into their I and Q parts and then segmented into non-overlapping time windows of 2048 samples which we call a *slice*. A signal correlation function is then run on each of the 2048 I and Q samples, each correlated with itself (I to I and Q to Q) to produce the auto-correlation at lags 0 to 2047. The resulting (2×4096) matrix emphasizes cyclostationary features, which are a key part of RF fingerprinting. This new slice contains a mirror image as a result of auto-correlation, so the first half (2×2048) is used as the modified feature set (i.e. slice) for training. Fig. 2 illustrates the preprocessing pipeline. In Section 6.4, we show that this preprocessing step to produce an auto-correlated representation of the data results in a substantially more effective detector than the raw IQ signal. On a current state-of-the-art laptop, converting a one second window of IQ data to this auto-correlated representation takes less than 70 milliseconds. This fast preprocessing makes it viable to deploy a trained HiNoVa detector in a time-critical setting.

4.2 Training

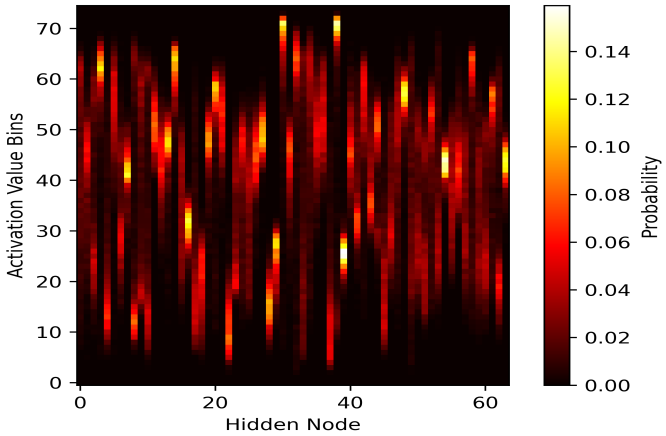
The architecture for the CNN+LSTM is shown in Table 2. We train the model with the ADAM optimizer [22] at a fixed learning rate (0.0001) with a cross-entropy loss function. We will discuss hyperparameter tuning in Section 6.1.

4.3 Detection

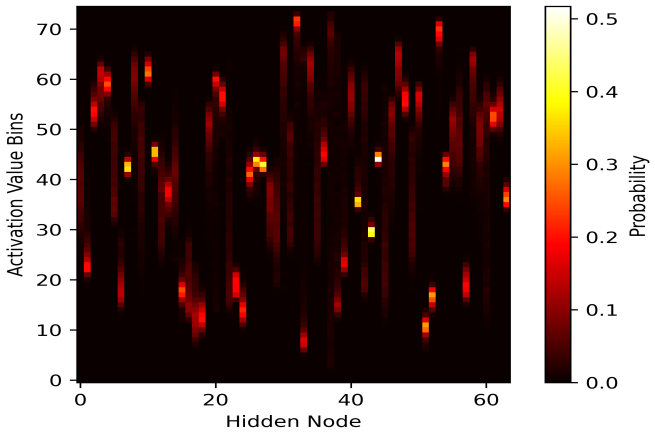
During the detection phase, the IQ data is preprocessed in the same way as in training. Each slice is passed through the trained CNN+LSTM and the final transition in the LSTM layer is extracted. The final transition was deter-

Table 2 – HiNoVa's CNN+LSTM architecture. Notation: Convolutional2D(*channels in*:*channels out*, *kernel dims*), BatchNorm2D(*num features*), MaxPool2D(*pool dims*), ReLU is a Rectified Linear Unit.

Input
↓
Convolutional2D (1:16, 2x256)
BatchNorm2D(16)
ReLU
Dropout(10%)
Convolutional2D (16:16, 2x256)
BatchNorm2D(16)
ReLU
Convolutional2D (16:32, 2x256)
BatchNorm2D(32)
ReLU
Dropout(10%)
Convolutional2D (32:32, 2x256)
BatchNorm2D(32)
ReLU
MaxPool2D(2x2)
LSTM(64)
Fully Connected Layer
LogSoftmax Layer
↓
Known or Unknown Device



(a) Device 5 fingerprint



(b) Device 6 fingerprint

Fig. 3 – Two unique fingerprints using HiNoVa under the wireless-Wi-Fi dataset (described in Section 5).

mined to be the most suitable for analysis due to the fact that at this point, the LSTM has processed all prior information within the slice. As a result, the internal nodes of the LSTM, specifically the forget gate and cell state, now contain both the long-term and short-term memory associated with the entire slice. This encoding effectively represents the transmission of the device during this specific time slice and is used to create a unique fingerprint.

4.4 Hidden state value fingerprinting

Algorithm 1 shows how HiNoVa uses the hidden state values within a trained CNN+LSTM to produce a unique fingerprint for each device in the training set. The first step involves aggregating, for each known device, the hidden node values from all the correctly classified slices during training. Then, for each known device, a histogram with B bins is built that describes the distribution of the hidden state values (i.e. h_t in Eqn *(6)*) for each hidden layer node in the LSTM. With M hidden state nodes, this histogram will be a $(M \times B)$ matrix, which serves as the unique fin- gerprint for that device. Examples of these fingerprints are shown in Fig. 3.

Algorithm 1 The Fingerprint Generation Algorithm

Require: H ▷ Hidden node values from correctly classified training slices

$FP \leftarrow zeroes(K_{known} \times M \times B)$
for $k \leftarrow 0$ **to** $(K_{known} - 1)$ **do** ▷Over known devices
  for $m \leftarrow 0$ **to** $(M - 1)$ **do** ▷ Over hidden nodes
    $H_{k,m} \leftarrow H[k, m]$
    $FP[k, m, :] \leftarrow$ Histogram($H_{k,m}$,B)
  end for
end for
return FP

1. K_{known}: the number of closed-set devices
2. M: the number of hidden nodes
3. B: the number of bins in the histogram
4. $Histogram(Values, B)$: Creates a histogram for $Values$ with B bins

4.5 Open-set fingerprint correlation

A number of different approaches can be used to compare test set device fingerprints to the fingerprints of the known devices. For instance, we could compute the probability of a test slice belonging to the histogram for that device, since the histogram is a valid probability distribution. We experimented with different approaches and found that correlations produced the best results. The most common approach for measuring correlation is Pearson's correlation coefficient, which makes a strong assumption that the relationship between two variables is linear. To avoid this strict assumption, we investigated Kendall's τ [23], which is a non-parametric measure of correlation that quantifies the rank-order association between two variables.

To compute Kendall's τ, let $fp_i = (fp_i^1, \dots, fp_i^{M*B})$ be the $M * B$ features (i.e. matrix values) for the fingerprint for the ith known device. Furthermore, let $fp_j = (fp_j^1, \dots, fp_j^{M*B})$ be the $M * B$ matrix values for the fingerprint of the jth device seen in the test set. Kendall's τ measures the rank correlation in terms of the ranks of the magnitudes of the features $(fp_i^1, \dots, fp_i^{M*B})$ and $(fp_j^1, \dots, fp_j^{M*B})$. Specifically, two feature indices $i1$ and $i2$ are said to be *concordant* if $fp_i^{i1} > fp_i^{i2}$ and $fp_j^{i1} > fp_j^{i2}$ (or equivalently if $fp_i^{i1} < fp_i^{i2}$ and $fp_j^{i1} < fp_j^{i2}$), otherwise they are said to be *discordant*. Computing Kendall's τ (see *(7)*) requires the number of concordant (P) and discordant pairs (Q), as well as the number of tied pairs of feature indices only in fp_i (T) and only in fp_j (U).

$$\tau = \frac{P - Q}{\sqrt{(P + Q + T) \cdot (P + Q + U)}} \qquad (7)$$

We chose Kendall's τ because it produced better performance than a linear correlation.

Algorithm 2 illustrates the unknown device detection process. Each test device has its slices converted to a test fingerprint, which is an $M \times B$ histogram. The test fingerprint for the kth test device was compared to all the known fingerprints, and its maximal rank correlation coefficient τ_k^* was computed. We use $(1 - \tau_k^*)$ to indicate the degree to which the test device was not correlated to a known device. If the value $(1 - \tau_k^*)$ was above a threshold, an open-set flag was raised.

5. TESTBED AND DATASETS

This work utilizes three RF datasets: LoRa, wireless-Wi-Fi, and wired-Wi-Fi which have been collected using a testbed of 15 PyCom IoT devices as transmitters [5, 24]: nine Fipy boards and six Lopy4 boards on top of PySense sensor shields (pictured in Fig. 4 (top)). Pycom devices are equipped with ESP32, Semtech SX1276, and Sequans Monarch chips that support Wi-Fi b/g/n, Bluetooth, LoRa, Sigfox, and Narrowband IoT network protocols.

On the reception side, we used an Ettus Universal Software Radio Peripheral (USRP) B210 with a VERT900 antenna for the data acquisition.

Algorithm 2 The Open-Set Detector

Require: FP ▷ Fingerprint Tensor (Alg. 1)
Require: $H_{test} \leftarrow \mathbf{K_{test}} \times \mathbf{M} \times \mathbf{S_{test}}$
Require: $FP_{test} \leftarrow zeroes(\mathbf{K_{test}} \times \mathbf{M} \times \mathbf{B})$
Require: $result \leftarrow zeroes(\mathbf{K_{test}})$
for $k \leftarrow 0$ **to** $(K_{test} - 1)$ **do**
  for $m \leftarrow 0$ **to** $M - 1$ **do**
    $FP_{test}[k, m, :] \leftarrow Histogram(H_{test}[k, m, :], B)$
  end for
end for
for $k \leftarrow 0$ **to** $(K_{test} - 1)$ **do**
  for $l \leftarrow 0$ **to** $(K_{known} - 1)$ **do**
    $\tau_{k,l} = KT(flatten(FP[l]), flatten(FP_{test}[k]))$
  end for
  $\tau_k^* = \max_l(\tau_{k,l})$
  $result[k] = (1 - \tau_k^*)$
end for
return result

1. K_{test}: the total number of test devices
2. M: the number of hidden nodes
3. S_{test}: the number of test slices per device
4. B: the number of bins in the histogram
5. H_{test}: the hidden state values for the test slices
6. FP_{test} : the test fingerprints
7. K_{known} : the number of known devices
8. KT: Kendall's τ correlation function
9. $flatten$: function to flatten 2D matrix to 1D vector
10. τ_k: The rank correlation coefficient for device k
11. $result$: the per-device vector of unthresholded predictions (higher is more indicative of an unknown device)

5.1 LoRa dataset

We transmitted LoRa transmissions using LoRa modulation, a proprietary physical layer implementation that utilizes a variant of the Chirp Spread Spectrum (CSS) technique [25]. This technology operates in the subGHz ISM band and strategically balances data rate with coverage range, power consumption, and/or link robustness. Each Pycom device was connected to a dedicated LoRa antenna, powered by a LiPo battery, and set up to transmit LoRa signals within the 915MHz US band, configured with the following parameters: Raw-LoRa mode, 125kHz bandwidth, a Spreading Factor (SF) of 7, a preamble of 8, a TX power of 20dBm, and a coding rate of 4/5. The devices were programmed to send the same LoRa messages for 20s in a round-robin fashion with a 15min gap between devices, generating 20M complex-valued samples per device. Positioned five meters from the receiver, these transmissions were captured by the USRP B210 receiver and sampled at a rate of 1 MSps. The datasets can be downloaded from NetSTAR Laboratory at http://research.engr.oregonstate.edu/hamdaoui/datasets. Detailed description of the LoRa dataset is provided in [5].

5.2 Wi-Fi dataset

We reprogrammed the same 15 Pycom devices to transmit Wi-Fi IEEE802.11B frames at a center frequency of 2.412GHz and a bandwidth of 20MHz. The data-capturing process was initiated 12 minutes after device activation to ensure the stability of the hardware [26]. The transmitters were programmed to consecutively transmit identical IEEE 802.11b frames, each lasting 559 microseconds, with a small gap in between. To eliminate any data dependency on the identity of the Wi-Fi transmitter, all transmitters were set to broadcast identical packets, featuring the same spoofed MAC address and a payload of zero bytes. For data acquisition, we employed an Ettus USRP B210 receiver, synchronized with an external signal synthesizer to enhance both sampling accuracy and stability. The power supply for all devices was facilitated through USB connections from an HP laptop. This setup and synchronization strategy were crucial in ensuring precise and stable data capture, providing a robust foundation for our subsequent analysis. The Wi-Fi frames have been sampled and digitally down-converted at a sample rate of 45MSps. Each Wi-Fi capture lasts for two minutes generating more than 5000 frames per device where each frame consists of 25, 170 complex-valued samples. While the transmitters were located 1m away from the receiver and connected to the same antenna in the wireless-Wi-Fi dataset, a 12inch SMA cable was used to connect them directly to the USRP receiver in the wired-Wi-Fi dataset as shown in Fig. 4 (right). Both of these datasets are publicly available at http://research.engr.oregonstate.edu/hamdaoui/datasets. Detailed descriptions of the Wi-Fi dataset are provided in [27, 28].

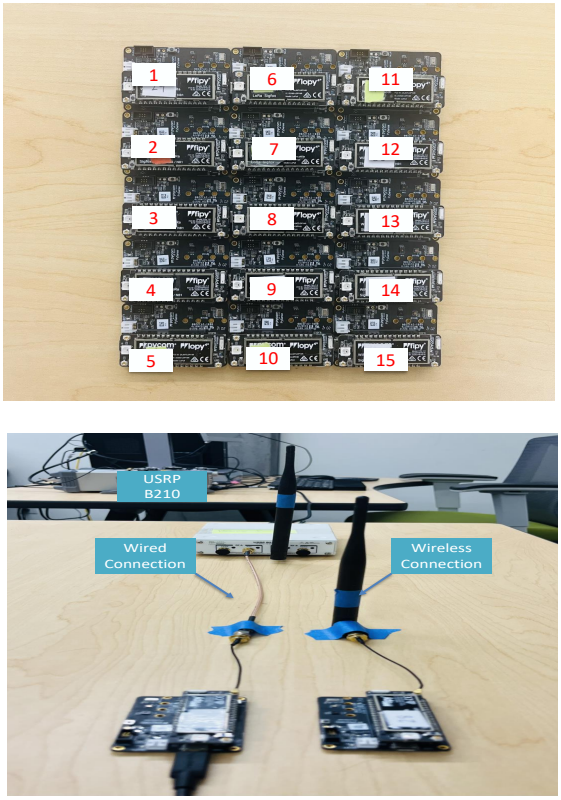


Fig. 4 – IoT testbed consisting of 15 Pycom transmitting devices (top) and a USRP B210 receiving device (bottom).

6. RESULTS AND DISCUSSION

For each of the three studied datasets, we set up three experiments in which we randomly selected 10 devices to be the known devices and five devices to be the unknown devices. We then evaluate our approach using a variant of 5-fold cross-validation designed to handle evaluation of open-set detection. We use a dataset with an equal number of data samples (i.e. slices) from each of the 15 devices. We divide each device's data into five non-overlapping equally-sized partitions. Under the traditional cross-validation process, in each fold of cross-validation, four of the partitions for that device are used as the training set while the remaining partition is used as the test set. The partitions are reassigned to training and testing in the other folds, such that each fold ends up using a different partition for testing, with no overlap between test sets for each fold. Data from the 10 known devices follow this traditional 5-fold cross-validation process. The main difference in our variant occurs with the test partition in each fold. In open-set detection, the test set contains both the test partition for the 10 known devices, as well as the test partition for the five unknown devices. We emphasize that in each fold, the data from the five unknown devices are only seen during testing and never seen during training.

Thus, to summarize the overall process, in each fold of cross-validation, HiNoVa is trained on the training set. After training, we generated 10 device fingerprints using the correctly classified samples from the 4 partitions of the known device training data. During the detection phase, HiNoVa takes each test sample from the test partition and compares it to the 10 known device fingerprints to perform a binary prediction as to whether or not the sample belongs to a known or unknown device.

6.1 Algorithms and performance metrics

We compare HiNoVa against a number of other open-set detection methods. These are summarized next:

1) CNN with MaxLogit (CNN Max Logit): This baseline uses a CNN augmented with the MaxLogit process for detecting open set instances. As was pointed out in a recent work [10], MaxLogit, though simple, is a strong open-set detector.

2) CNN+LSTM with MaxLogit (CNN+LSTM Max Logit): The previous baseline interprets each observation in a slice as an i.i.d. data instance. In reality, the observations in a slice have a sequential relationship and using a CNN+LSTM instead of a CNN enables the detector to model these sequential relationships. As before, we use the MaxLogit approach for open-set detection.

3) OpenMax [29] (CNN+LSTM OpenMax): This baseline re-weights the activation vectors that go into the final Softmax layer to better separate the known from the unknown devices. The weighting function uses a Weibull distribution for modeling extreme values and is used in OpenMax to model the right tail of the activation distribution corresponding to the highest activation values. OpenMax only re-weights the activations for the top α classes with the highest activation values.

4) Akar et al. [13] (Akar): This algorithm is the most closely related work to our approach as it is an open-set detector specifically for time series. We refer to this approach as *Akar*. The Akar algorithm uses Dynamic Time Warping (DTW) to compute the similarity between a test set time series and the barycenters of known devices. Devices that are greater than a specified threshold in terms of DTW distance to the barycenters of known devices or less than a specified threshold in terms of cross-correlation are declared to be open set devices.

We use the Area Under Precision-Recall Curve (AUPRC as the evaluation metric [30] since there is a significant class imbalance as we have twice as much data from known devices than from unknown devices during testing. AUPRC considers the trade-off between precision and recall across a range of detection thresholds and yields an overall threshold-independent summary statistic of the detector's performance.

To determine the hyperparameter settings for our deep learning models, we use post-hoc tuning on CNN+LSTM MaxLogit. We use CNN+LSTM MaxLogit because parts of its architecture are shared with CNN MaxLogit and CNN+LSTM OpenMax. Post-hoc tuning refers to looking at the performance of CNN+LSTM MaxLogit on the test set, which gives CNN+LSTM MaxLogit an unfair advantage as it is allowed to see the test set, but we will show that even with this advantage, HiNoVa still significantly outperforms the MaxLogit models.

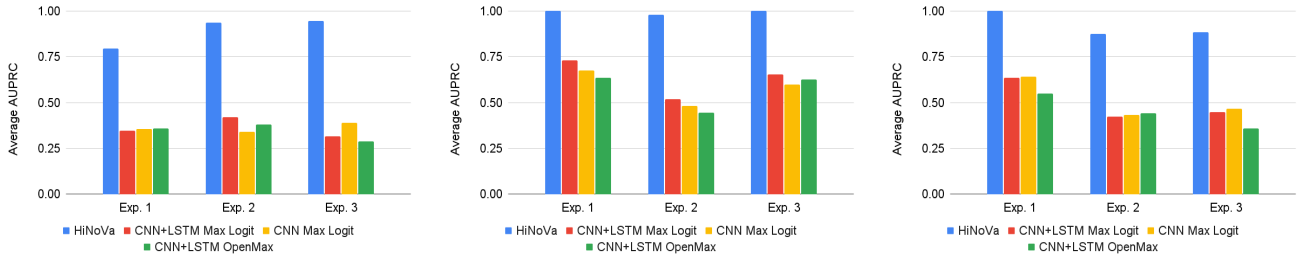
Specifically, we post-hoc tune the kernel size (2×256) and dropout rate (10%) in the CNN layer to achieve high accuracy in closed set classification using a grid search. Attaining good closed set accuracy has recently been shown to produce good open set detectors [10]. We also post-hoc tune the number of hidden nodes to achieve high AUPRC for the open-set prediction task for CNN+LSTM MaxLogit. The resulting values of these hyperparameters were applied to HiNoVa, which clearly puts it at a disadvantage because these hyperparameters were tuned for a completely different algorithm (i.e. CNN+LSTM MaxLogit), but HiNoVa still performs well.

We evaluated HiNoVa with 25, 50, 75 and 100 bins and found that it resulted in small differences in AUPRC (< 0.03). As a result, we report results with 25 bins in our experiments.

6.2 Comparison results

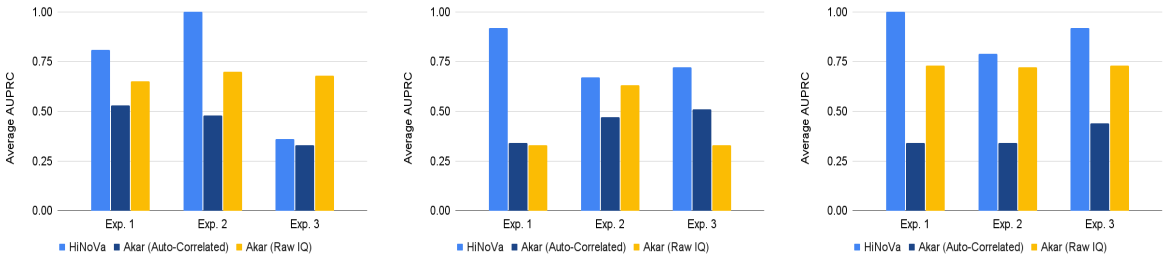
We begin by comparing HiNoVa against CNN+LSTM Max Logit, CNN Max Logit and CNN+LSTM OpenMax. Our evaluation is done using three different RF datasets: LoRa, wireless-Wi-Fi, and wired-Wi-Fi, as described in Section 5. Fig. 5 shows the average AUPRC values for the LoRa, wireless-Wi-Fi, and wired-Wi-Fi datasets respectively. HiNoVa consistently outperformed the other methods, achieving statistically significant improvements (Wilcoxon Signed Rank Test, $\alpha = 0.05$) in all three experiments. CNN+LSTM MaxLogit, CNN MaxLogit, and OpenMax lagged behind both HiNoVa by a substantial gap in AUPRC, with no consistent top performer in this second tier of algorithms.

Overall, the results suggest that HiNoVa is an effective detector of unknown devices using LoRa, wireless-Wi-Fi and wired-Wi-Fi protocols, outperforming other methods by a significant margin. The hidden state values correspond to a compact representation of the autocorrelation lags in the IQ data within a slice, and the distribution of this representation, as represented in the histogram used to derive the fingerprint, provides an effective summary of the device-specific information that HiNoVa is able to leverage. Finally, the MaxLogit and OpenMax approaches only



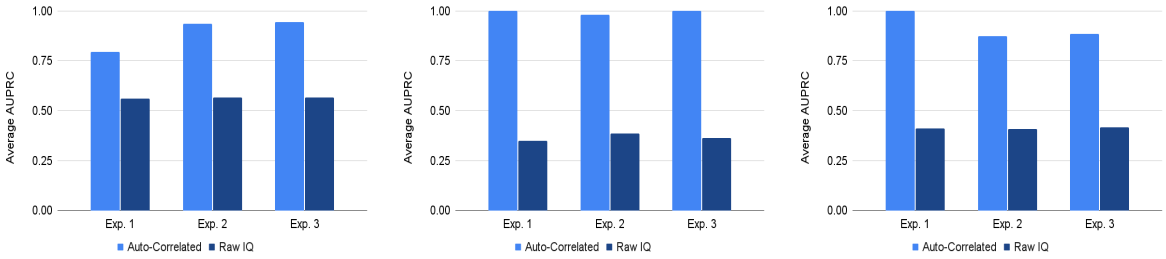
(a) LoRa **(b)** Wireless-Wi-Fi **(c)** Wired-Wi-Fi

Fig. 5 – Average test AUPRC for `HiNoVa` vs other algorithms on the (a) LoRa, (b) wireless-Wi-Fi and (c) wired-Wi-Fi datasets. The differences between `HiNoVa` and the other algorithms are statistically significant (Wilcoxon Signed Rank Test, $\alpha = 0.05$).



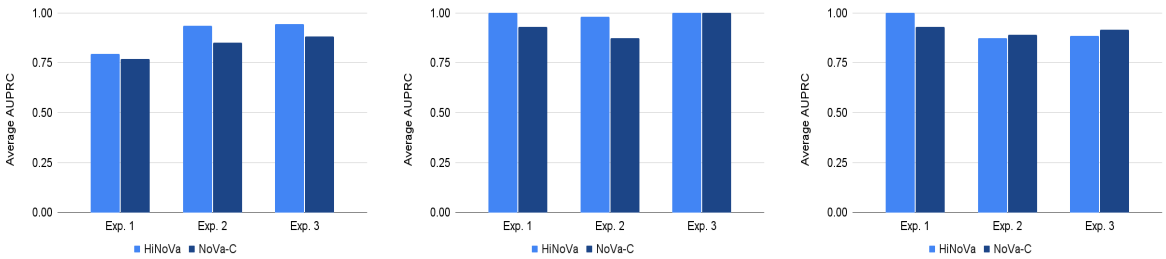
(a) LoRa dataset **(b)** Wireless-Wi-Fi dataset **(c)** Wired-Wi-Fi dataset

Fig. 6 – Test AUPRCs for HiNoVa vs Akar (Auto-Correlated) and Akar (Raw IQ).



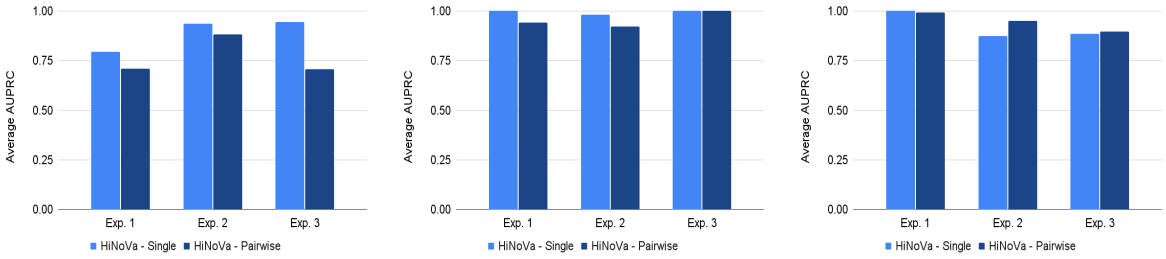
(a) LoRa dataset **(b)** Wireless-Wi-Fi dataset **(c)** Wired-Wi-Fi dataset

Fig. 7 – Average test AUPRCs for `HiNoVa` using Auto-Correlated data vs Raw IQ data.



(a) LoRa dataset **(b)** Wireless-Wi-Fi dataset **(c)** Wired-Wi-Fi dataset

Fig. 8 – Average test AUPRCs for HiNoVa vs HiNoVa-C.



(a) LoRa dataset **(b)** Wireless-Wi-Fi dataset **(c)** Wired-Wi-Fi dataset

Fig. 9 – Average test AUPRCs for the single hidden node detector vs the pairwise hidden node detector.

rely on the logits of the penultimate layer of the NN. These logits, which are used to derive the output probabilities from the NN, lack the information contained in the fingerprints and are thus less effective at identifying unknown devices.

6.3 Comparison with Akar et al. [13]

For the Akar algorithm, we used the implementation provided by the authors themselves[1]. The Akar algorithm is computationally expensive as its use of DTW as a distance metric is quadratic in the length of the input time series. Even with efficient DTW methods in their implementation, the algorithm is prohibitively slow for the data in our experiments from Section 6.2, taking weeks to complete. As such, we modified our experimental setup in Section 6.2 in the following ways. First, we truncate the training data to be 25% of its original size. Second, we reduce the number of known devices to six and the number of unknowns to three. Finally, we report AUPRC over one-fold of the 5-fold cross-validation (over the truncated training data size) for our three datasets. We also modified the Akar algorithm to output a binary classification of *known* or *unknown*, rather than classifying the device as one of K known devices or a $(K+1)$th value of *unknown*. This modification to a binary classification was necessary to allow direct comparison to `HiNoVa` in terms of the AUPRC metric. We apply the Akar algorithm to both the auto-correlation representation and the raw IQ version of the data.

The Akar algorithm also requires tuning of the α and β hyperparameters which are used to set the thresholds for the DTW distance (τ_k^{dist}) and the cross-correlation (τ_k^{cc}) for device k respectively:

$$\tau_k^{dist} = \tilde{\mu}_k^{dist} + \alpha \cdot \sigma_k^{dist} \quad (8)$$

$$\tau_k^{cc} = \tilde{\mu}_k^{cc} - \beta \cdot \sigma_k^{cc} \quad (9)$$

Here, α and β define the multiples of standard deviations beyond the means ($\tilde{\mu}_k^{dist}$ and $\tilde{\mu}_k^{cc}$) such that a data instance exceeding this threshold is declared an unknown device. The source code for the original algorithm performs a grid search over the training set for values of α and β and then uses the best configuration of hyperparameter values on the test set. To speed up the experiments, we perform *post-hoc* tuning on the Akar algorithm, which evaluates the same combinations of (α, β) used in the source code grid search but we report the best setting of these hyperparameters on the *test* data instead of the training data. By using the test data, this post-hoc tuning reports the actual best performing values of the hyperparameters for the Akar algorithm on the test data, giving it an advantage over `HiNoVa`.

We also run `HiNoVa` on the auto-correlated representation of these same training datasets and report results on the same single fold of cross-validation. We use the hyperparameters for `HiNoVa` selected in the experiments from Section 6.2.

Fig. 6 depicts the difference in AUPRC between `HiNoVa` and the Akar variants. In general, the Akar algorithm performs better using the raw IQ representation than the auto-correlation representation. However, `HiNoVa` outperforms Akar, applied to the raw IQ representation, in 8 out of 9 experiments, often by a substantial amount. Due to only a single fold being run on each experiment, we do not have a large enough sample size to reliably perform a hypothesis test to establish statistical significance. The use of DTW by the Akar algorithm is largely focused on matching the overall shape of the input data and is thus unable to learn an effective representation of the input data that captures salient features for detecting unknown devices. We also note that even with the modifications to the experimental setup, the runtime of Akar's algorithm is still impractical for real-time deployment as each experiment took several days to complete.

6.4 Auto-correlated vs raw IQ

We demonstrate the value of applying a preprocessing step that converts the 2048 sample window of raw IQ data into the auto-correlated representation described in Section 4.1. The auto-correlated representation reduces

[1] https://github.com/tolgaakar/Open-Set-Recognition-for-Time-Series-Classification

noise and allows the deep learning layers to use features related to auto-correlation in its prediction. This preprocessing step can be run efficiently on large datasets without significantly increasing system run-time; in our implementation, we apply Python's correlate function which is an efficient matrix operation. We compare the results of applying HiNoVa to the raw IQ values versus the auto-correlated representation using the same experimental setup as in Section 6.2. Fig. 7 illustrates the large improvement in AUPRC (0.20 to 0.65) with using the auto-correlated representation versus the raw IQ data. These improvements are statistically significant (Wilcoxon Signed Rank Test, $\alpha = 0.05$).

6.5 HiNoVa vs HiNoVa-C

A natural variant of the `HiNoVa` algorithm is to use the cell state of the LSTM instead of the hidden state. Historically, in a recurrent neural network, which is a pre-cursor to the LSTM, the hidden state corresponds to the short term memory as it remembers the state in the previous time step. In an LSTM, the cell state was introduced as a longer term memory of patterns that extend beyond the previous time step. However, in comparing equations *(5)* and *(6)*, the cell state c_t misses any mathematical operation on the output gate, so intuitively, the hidden state contains some information that the cell state does not.

Using the same experimental setup as the beginning of Section 6.2, we repeated the experiments with a version of `HiNoVa` called `HiNoVa-C`, which builds the fingerprints using patterns in the cell state rather than the hidden state. Fig. 8 illustrates the results. For six experiments (All LoRa experiments, experiments 1 and 2 for wireless-Wi-Fi and experiment 1 of wired-Wi-Fi), `HiNoVa` outperforms `HiNoVa-C` while in two experiments, the differences are much closer, with a slight improvement for `HiNoVa-C` in experiments 2 and 3 of the wired-Wi-Fi dataset. These results suggest that the hidden state has a slight advantage when building device fingerprints, but more investigation is needed to determine which parts of the internal state of a LSTM would be best for RF fingerprinting.

6.6 Pairwise vs single hidden node values

Since LSTMs use the hidden node value from the previous time step (h_{t-1}) to compute the value of the current hidden node (h_t), we explore building the RF fingerprint with the pair of hidden node values at consecutive times (h_{t-1}, h_t) instead of the hidden node value at a single time (h_t). Fig. 9 compares the performance of a single vs pairwise hidden node value detector. Fig. 9 shows that for `HiNoVa`, the results are mixed, with a pairwise detector outperforming the single node detector in about half of the experiments. These results indicate that pairwise transitions can have predictive value in some cases, but in other cases they are simply noise. Given the additional computational cost of the pairwise node detector in both time and memory, we recommend using the single node detector.

7. CONCLUSION

`HiNoVa` is a novel open-set detection method based on the activation patterns of the hidden states within a CNN+LSTM model. This approach significantly improves the AUPRC on LoRa, wireless-Wi-Fi, and wired-Wi-Fi datasets over other open-set detection methods. Additionally, because of its structure, the proposed method can run on standard consumer hardware with minimal setup data and training time. Future work will investigate using attention-based deep learning models.

ACKNOWLEDGEMENTS

This work is supported in part by Intel/NSF Award No. 2003273.

REFERENCES

[1] S. Mathur, A. Reznik, C. Ye, R. Mukherjee, A. Rahman, Y. Shah, W. Trappe, and N. Mandayam. "Exploiting the physical layer for enhanced security [Security and Privacy in Emerging Wireless Networks]". In: *IEEE Wireless Communications* 17.5 (2010), pp. 63–70. DOI: 10.1109/MWC.2010.5601960.

[2] A. Elmaghbub and B. Hamdaoui. "LoRa Device Fingerprinting in the Wild: Disclosing RF Data-Driven Fingerprint Sensitivity to Deployment Variability". In: *IEEE Access* 9 (2021), pp. 142893–142909. DOI: 10.1109/ACCESS.2021.3121606.

[3] B. Hamdaoui and A. Elmaghbub. "Deep-learning-based device fingerprinting for increased LoRa-IoT security: Sensitivity to network deployment changes". In: *IEEE network* 36.3 (2022), pp. 204–210.

[4] B. Hamdaoui, N. Basha, and K. Sivanesan. "Deep Learning-Enabled Zero-Touch Device Identification: Mitigating the Impact of Channel Variability Through MIMO Diversity". In: *IEEE Communications Magazine* 61.6 (2023), pp. 80–85.

[5] A. Elmaghbub and B. Hamdaoui. "Comprehensive RF Dataset Collection and Release: A Deep Learning-Based Device Fingerprinting Use Case". In: *2021 IEEE Globecom Workshops (GC Wkshps)*. 2021, pp. 1–7. DOI: 10.1109/GCWkshps52748.2021.9682024.

[6] L. Puppo, W-K. Wong, B. Hamdaoui, and A. Elmaghbub. "HiNoVa: A Novel Open-Set Detection Method for Automating RF Device Authentication". In: *IEEE Symposium on Computers and Communications* (2023), pp. 1122–1128.

[7] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult. "Toward Open Set Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.7 (2013), pp. 1757–1772. DOI: 10.1109/TPAMI.2012.256.

[8] C. Geng, S. Huang, and S. Chen. "Recent Advances in Open Set Recognition: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.10 (Oct. 2021), pp. 3614–3631. DOI: 10.1109/TPAMI.2020.2981604.

[9] D. Hendrycks and K. Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: *5th International Conference on Learning Representations, ICLR 2017*. 2017.

[10] S. Vaze, K. Han, A. Vedaldi, and A. Zisserman. "Open-Set Recognition: A Good Closed-Set Classifier is All You Need". In: *International Conference on Learning Representations*. 2022.

[11] T. G. Dietterich and A. Guyer. "The familiarity hypothesis: Explaining the behavior of deep open set methods". In: *Pattern Recognition* 132 (Dec. 2022), p. 108931.

[12] Y. Sun, C. Guo, and Y. Li. "React: Out-of-distribution detection with rectified activations". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 144–157.

[13] T. Akar, T. Werner, V. K. Yalavarthi, and L. Schmidt-Thieme. "Open set recognition for time series classification". In: *Advances in Knowledge Discovery and Data Mining: 26th Pacific-Asia Conference, Part II*. Springer. 2022, pp. 354–366.

[14] 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)*. IEEE Press, 2019, pp. 1–10. DOI: 10.1109/DySPAN.2019.8935862.

[15] S. Hanna, S. Karunaratne, and D. Cabric. "Deep Learning Approaches for Open Set Wireless Transmitter Authorization". In: *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. 2020, pp. 1–5. DOI: 10.1109/SPAWC48557.2020.9154254.

[16] J. Gaskin, B. Hamdaoui, and W-K. Wong. "Tweak: Towards portable deep learning models for domain-agnostic LoRa device authentication". In: *arXiv preprint arXiv:2209.00786* (2023).

[17] J. Gaskin, A. Elmaghbub, B. Hamdaoui, and W-K. Wong. "Deep Learning Model Portability for Domain-Agnostic Device Fingerprinting". In: *IEEE Access* (2023).

[18] S. Karunaratne, S. Hanna, and D. Cabric. "Open Set RF Fingerprinting using Generative Outlier Augmentation". In: *2021 IEEE Glob. Commun. Conf.* 2021, pp. 01–07. DOI: 10.1109/GLOBECOM46510.2021.9685335.

[19] V. Chandola, A. Banerjee, and V. Kumar. "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41.3 (July 2009). DOI: 10.1145/1541880.1541882.

[20] S. Hanna, S. Karunaratne, and D. Cabric. "Open set wireless transmitter authorization: Deep learning approaches and dataset considerations". In: *IEEE Trans. Cogn. Commun. Netw.* 7.1 (2020), pp. 59–72.

[21] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9 (1997), pp. 1735–1780.

[22] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *3rd Int'l Conf. on Learning Representations, ICLR 2015*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[23] M. Kendall. "A New Measure of Rank Correlation". In: *Biometrika* 30.(1-2) (1938), pp. 81–89.

[24] A. Elmaghbub and B. Hamdaoui. "A Needle in a Haystack: Distinguishable Deep Neural Network Features for Domain-Agnostic Device Fingerprinting". In: *2023 IEEE Conference on Communications and Network Security (CNS)*. IEEE. 2023.

[25] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley. "A study of LoRa: Long range & low power networks for the internet of things". In: *Sensors* 16.9 (2016), p. 1466.

[26] A. Elmaghbub and B. Hamdaoui. "EPS: distinguishable IQ data representation for domain-adaptation learning of device fingerprints". In: *arXiv preprint arXiv:2308.04467* (2023).

[27] A. Elmaghbub, B. Hamdaoui, and W-K. Wong. "ADL-ID: Adversarial Disentanglement Learning for Wireless Device Fingerprinting Temporal Domain Adaptation". In: *IEEE International Conference on Communication 2023 - Mobile and Wireless Network Symposium*, pp. 6199–6204.

[28] B. Johnson and B. Hamdaoui. "On the Domain Generalizability of RF Fingerprints Through Multifractal Dimension Representation". In: *IEEE Conference on Communications and Network Security (CNS)*. IEEE. 2023, pp. 1–9.

[29] A. Bendale and T. E. Boult. "Towards Open Set Deep Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition*. Los Alamitos, CA, USA: IEEE Computer Society, 2016, pp. 1563–1572.

[30] S. Takaya and M. Rehmsmeier. "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets". In: *PloS one* 10.3 (2015), e0118432.

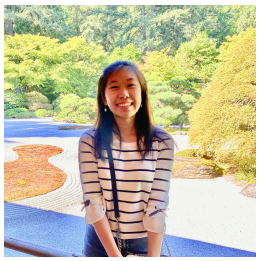
AUTHORS

Luke Puppo earned his B.S. (2021) and M.S. (2023) in computer science from Oregon State University. His interests are in data science, computer vision, machine learning, and generative AI. He currently works in industry dealing with time-series financial data.

Weng-Keen Wong received his Ph.D. and M. S. degrees in computer science from Carnegie Mellon University in 2004 and 2001 respectively. He received his B.Sc. degree from the University of British Columbia in 1997. He is currently a professor in the School of Electrical Engineering and Computer Science at Oregon State University. From 2016-2018, he served as a program director at the National Science Foundation under the Robust Intelligence Program in the Division of Information and Intelligent Systems. His research areas are in data mining and machine learning, with specific interests in anomaly detection, deep learning, probabilistic graphical models, computational sustainability and human-in-the-loop learning. He has authored over 70 papers in international journals and conferences.

Bechir Hamdaoui is a professor in the School of Electrical Engineering and Computer Science at Oregon State University. He received M.S. degrees in both electrical and computer engineering (2002) and computer science (2004), and a Ph.D. degree in electrical and computer engineering (2005) all from the University of Wisconsin-Madison. His research interests are in the general areas of intelligent networked systems, with a current focus on the intersection of applied AI, wireless, and security. He won several awards, including the ISSIP 2020 Distinguished Recognition Award, the ICC 2017 Best Paper Award, the IWCMC 2017 Best Paper Award, the 2016 EECS Outstanding Research Award, and the 2009 NSF CAREER Award. He serves/served as an associate editor for several journals, including IEEE Transactions on Mobile Computing, IEEE Transactions on Wireless Communications, IEEE Network, and IEEE Transactions on Vehicular Technology. He also chaired/co-chaired many IEEE conference programs/symposia, including the 2021 IEEE GLOBECOM testbeds4wireless workshop, the 2017 INFOCOM Demo/Posters program, the 2016 IEEE GLOBECOM Mobile and Wireless Networks symposium, and many others. He served as the chair of the IEEE Communications Society's Wireless Communication Technical Committee (WTC) for 2021 and 2022, and as a distinguished lecturer for the IEEE Communication Society for 2016 and 2017.

Abdurrahman Elmaghbub received the B.S. degree with summa cum laude, and MS in electrical and computer engineering from Oregon State University in 2019 and 2021, respectively, and is currently pursuing his Ph.D. degree in the School of Electrical Engineering and Computer Science at Oregon State University. His research interests are in the area of wireless communication and networking security with a current focus on applying deep learning to wireless device classification.



Lucy Lin is currently in her fourth year pursuing her Honors B.Sc. degree in computer science at Oregon State University. She is an undergraduate research assistant in Dr. Weng-Keen Wong's lab. Her research interests are in machine learning and computer vision.