HiNoVa: A Novel Open-Set Detection Method for Automating RF Device Authentication

Luke Puppo, Weng-Keen Wong, Bechir Hamdaoui, Abdurrahman Elmaghbub
School of EECS, Oregon State University, Corvallis, OR, USA
{puppol, wongwe, hamdaoui, elmaghba}@oregonstate.edu

***Abstract*—New capabilities in wireless network security have been enabled by deep learning, which leverages patterns in radio frequency (RF) data to identify and authenticate devices. Open-set detection is an area of deep learning that identifies samples captured from new devices during deployment that were not part of the training set. 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. We introduce a novel open-set detection approach based on the patterns of the hidden state values within a Convolutional Neural Network Long Short-Term Memory model. Our approach greatly improves the Area Under the Precision-Recall Curve on LoRa, Wireless-WiFi, and Wired-WiFi datasets, and hence, can be used successfully to monitor and control unauthorized network access of wireless devices.**

***Index Terms*—Device authentication; RF device fingerprinting; open-set detection; deep learning.**

I. 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].

RF fingerprinting is a recognized key method to enhance security in IoT networks [2]. 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 WiFi protocols. LoRa is a wireless communication technology designed for IoT devices that operates in the sub-gigahertz frequency range, enabling long-range, low-power, bi-directional 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 WiFi, 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 WiFi, has its practical use and is commonly adopted by various transmitters, and hence, our proposed open-set detection method is tested using both protocols.

A. Open-Set Detection and Device Authentication

Supervised machine learning 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 wireless device fingerprinting is used for security purposes. In this security use case, the classes correspond to *known* devices and it is crucial for the system to accurately detect *unknown* devices (i.e. the open-set devices) to raise an alert. For this type of problems, *Open-set* detection [3] 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.

B. Related Work

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 [4]. In a 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 [5], [6] 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 [5]. 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 [7] analyzes the internal activations of neural networks and identifies highly distinctive signature patterns for open-set distributions. Dietterich et al. [6] 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. [8], 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. [9] use the maximum 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. [10] investigate a variety of methods such as the maximum softmax probability and methods that incorporate data from known unauthorized devices. Gaskin et al. [11] propose Tweak, a lightweight calibration approach that leverages metric learning to achieve high open-set accuracy without the need for model re-training, making it more suitable for resource-constrained applications. In a recent work, Karunaratne et al. [12] 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 time series nature of the data with a CNN+LSTM and performing open-set detection. Another closely related area to open-set detection is anomaly detection [13]. In anomaly detection, the goal is to identify individual outliers that are rare with respect to the ”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 ”normal” 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 generally outperform anomaly detection methods for detecting unknown devices [14].

C. Contributions

We introduce `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 II presents the machine learning architecture used by our method. Section III presents the details of the `HiNoVa` algorithm. Section IV describes the LoRa, Wireless-WiFi, and Wired-WiFi datasets used in our evaluation and Section V evaluates the performance of `HiNoVa` using these datasets. The last section concludes the paper.

II. The 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 $\mathbf{h}_t$ and an output vector $\mathbf{y}_t$ at each time step t. The vector $\mathbf{h}_t$ depends not only on the input vector $\mathbf{x}_t$ at time step t, but also on the hidden state vector $\mathbf{h}_{t-1}$ 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) [15] layer was developed to handle long-term dependencies in the input sequence more effectively.

A. Long-Short-Term Memory (LSTM) Layer

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

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{ii}\mathbf{x}_t + \mathbf{b}_{ii} + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_{hi}) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{if}\mathbf{x}_t + \mathbf{b}_{if} + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_{hf}) \\
\mathbf{g}_t &= \tanh(\mathbf{W}_{ig}\mathbf{x}_t + \mathbf{b}_{ig} + \mathbf{W}_{hg}\mathbf{h}_{t-1} + \mathbf{b}_{hg}) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{io}\mathbf{x}_t + \mathbf{b}_{io} + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_{ho}) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{1}
$$

Each term in the LSTM equations is described below:

- $\mathbf{x}_t$: The input vector at time t.
- $\mathbf{h}_{t-1}$: 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: 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 is updated based on the input gate i_t, forget gate f_t, and cell gate g_t. The input gate controls how much new information is added to the cell state and the forget gate controls how much old information is retained. The cell gate controls what new information is added to the cell state, by applying an activation function (i.e. tanh) to the input and previous hidden state.

Finally, the output gate o_t controls how much of the current cell state is output as the new hidden state h_t. The new hidden state 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 the ideal choice for a unique latent description.

B. Convolutional Neural Network LSTMs (CNN+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.

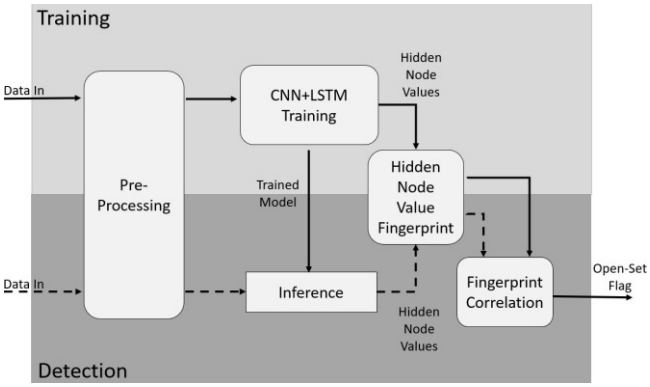


Fig. 1: The proposed ML architecture of HiNoVa.

III. Methodology

Figure 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.

A. Pre-Processing

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 an USRP receiver and saved in a complex number format.

To pre-process 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.

B. Training

The architecture for the CNN+LSTM is shown in Table I. We train the model with the ADAM optimizer at a fixed learning rate (0.0001) with a cross-entropy loss function. We will discuss hyper-parameter tuning in Section V-A.

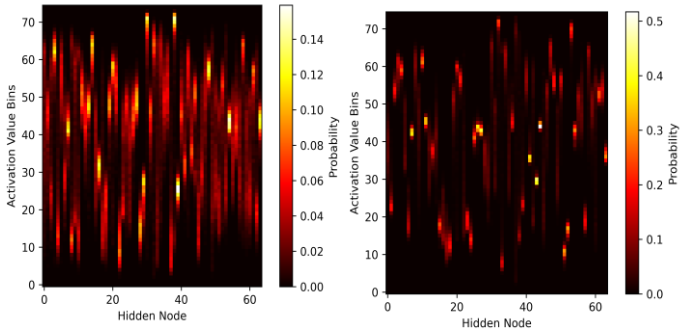
C. Detection

During the detection phase, the IQ data is pre-processed 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 determined to be the most suitable for analysis due to the fact that at this point, the

TABLE I

HiNoVa's CNN+LSTM architecture. Notation: Conv2d(*channels in*:*channels out*, *kernel dims*), BNorm2D(*num features*), MaxPool2d(*pool dims*)

Layer
Conv2d (1:16, 2x256) ➔ BNorm2d(16) ➔ ReLU ➔ Dropout(10%)
Conv2d (16:16, 2x256) ➔ BNorm2d(16) ➔ ReLU
Conv2d (16:32, 2x256) ➔ BNorm2d(32) ➔ ReLU ➔ Dropout(10%)
Conv2d (32:32, 2x256) ➔ BNorm2d(32) ➔ ReLU ➔ MaxPool2d(2x2)
LSTM(64) ➔ Fully Connected ➔ LogSoftmax



(a) Device 5 Fingerprint (b) Device 6 Fingerprint

Fig. 2: Two unique fingerprints using HiNoVa under the Wireless-WiFi Dataset (described in Sec. IV).

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.

D. 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 (1)) for each hidden layer node in the LSTM. With M hidden state nodes, this histogram will be a (M $\times$ B) matrix for each device, which serves as the unique fingerprint for that device. Examples of these fingerprints are shown in Fig. 2.

E. 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

Algorithm 1 The Fingerprint Generation Algorithm

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

```
FP ← zeroes(K_known × M × B)
for k ← 0 to (K_known − 1) do        ▷ Over known devices
    for m ← 0 to (M − 1) do          ▷ Over hidden nodes
        H_{k,m} ← H[k, m]
        FP[k, m, :] ← 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

strong assumption that the relationship between two variables is linear. To avoid this strict assumption, we investigated Kendall's τ [16], which is a non-parametric measure of correlation that quantifies the rank-order association between two variables.

To compute Kendall's τ, let $\boldsymbol{fp_i} = (fp_i^1, \ldots, fp_i^{M*B})$ be the $M * B$ features (i.e. matrix values) for the fingerprint for the ith known device. Furthermore, let $\boldsymbol{fp_j} = (fp_j^1, \ldots, fp_j^{M*B})$ 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 $(fp_i^1, \ldots, fp_i^{M*B})$ and $(fp_j^1, \ldots, fp_j^{M*B})$. 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 (2)) requires the number of concordant (P) and discordant pairs (Q), as well as the number of tied pairs of feature indices only in $\boldsymbol{fp_i}$ (T) and only in $\boldsymbol{fp_j}$ (U).

$$\tau = \frac{P - Q}{\sqrt{(P + Q + T) \cdot (P + Q + U)}} \tag{2}$$

We chose Kendall's τ because it produced significantly 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.

IV. Testbed and Datasets

In this work, we utilized three RF datasets: LoRa, Wireless-WiFi, and Wired-WiFi which have been collected using a testbed of 15 PyCom IoT devices as transmitters: 9 Fipy boards and 6 Lopy4 boards on top of PySense sensor shields (pictured

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 Tau 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)

in Fig. 3 (left)). On the reception side, we used an Ettus USRP (Universal Software Radio Peripheral) B210 with a VERT900 antenna for the data acquisition. For the LoRa dataset, we captured the LoRa transmissions of a duration of 20s each, in an indoor environment where the devices were located 5m away from the receiver. Each Pycom device was connected to a dedicated LoRa antenna and configured to transmit LoRa transmissions at the 915MHz and 125KHz bandwidth. These transmissions have been sampled by the USRP receiver at a rate of 1MSps. Refer to the Indoor LoRa dataset section in [2], [17] for more details.

For the WiFi datasets, the same Pycom devices were programmed to transmit WiFi IEEE802.11B frames at a center frequency of 2.412GHz and 20MHz bandwidth. These frames have been sampled and digitally down-converted by the same USRP receiver at a sample rate of 45MSps. Each WiFi capture lasts for 2 minutes generating more than 5000 frames per device where each frame consists of 25170 complex-valued samples. While the transmitters were located 1m away from the receiver and connected to the same antenna in the wireless WiFi dataset, a 12inch SMA cable was used to connect them directly to the USRP receiver in the wired WiFi dataset as shown in Fig. 3 (right).

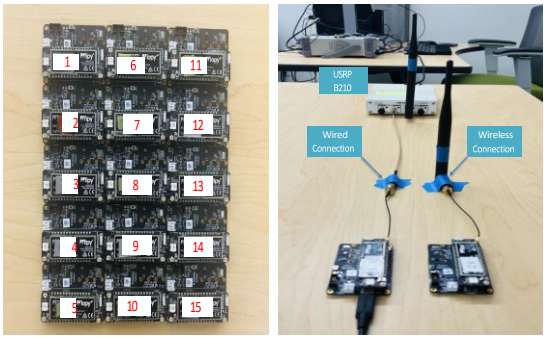


Fig. 3: IoT Testbed consisting of 15 Pycom transmitting devices (left) and a USRP B210 receiving device (right).

V. Results and Discussion

For each of the three studied datasets, we set up 3 experiments in which we randomly selected 10 devices to be the known devices and 5 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 5 non-overlapping equally-sized partitions. Under the traditional cross-validation process, in each fold of cross-validation, 4 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 5 unknown devices. We emphasize that in each fold, the data from the 5 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.

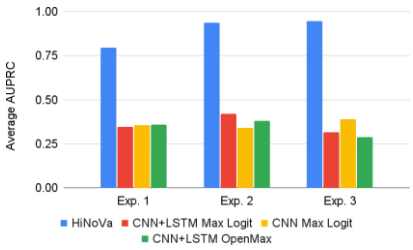
A. Algorithms and Performance Metrics

We compare `HiNoVa` against a number of other open-set detection methods:

1) CNN model using MaxLogit: This baseline uses a CNN augmented with the MaxLogit process for detecting open set instances. As was pointed out in a recent work [5], MaxLogit, though simple, is a strong open-set detector.

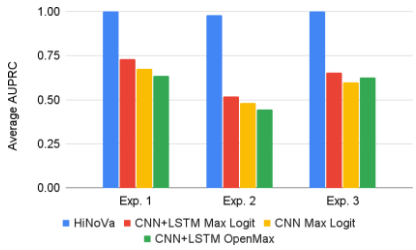
TABLE II

Average Test AUPRC for HiNoVa vs. other algorithms on the (a) LoRa, (b) Wireless-WiFi and (c) Wired-WiFi datasets. The graphs on top plot the results in the tables below. Statistical significance is indicated with * in the tables.



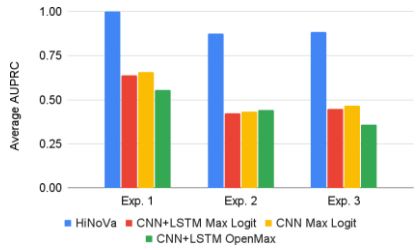
	Exp. 1	Exp. 2	Exp. 3
HiNoVa	**0.80***	**0.94***	**0.95***
CNN+LSTM/ MaxLogit	0.35	0.42	0.32
CNN/MaxLogit	0.36	0.34	0.39
CNN+LSTM/ OpenMax	0.36	0.38	0.29

(a) LoRa



	Exp. 1	Exp. 2	Exp. 3
HiNoVa	**1.00***	**0.98***	**1.00***
CNN+LSTM/ MaxLogit	0.73	0.52	0.65
CNN/MaxLogit	0.68	0.48	0.60
CNN+LSTM/ OpenMax	0.63	0.45	0.63

(b) Wireless-WiFi



	Exp. 1	Exp. 2	Exp. 3
HiNoVa	**1.00***	**0.87***	**0.89***
CNN+LSTM/ MaxLogit	0.59	0.43	0.45
CNN/MaxLogit	0.63	0.43	0.47
CNN+LSTM/ OpenMax	0.56	0.44	0.36

(c) Wired-WiFi

2) CNN+LSTM model using MaxLogit: 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 [18]: This baseline reweights the activation vectors that go into the final Softmax layer to better separate the known from unknown devices. The weighting function is based on a Weibull distribution, which is used to model extreme values and is used in OpenMax to model the right tail of the activation distribution corresponding to the highest activation values. OpenMax only reweights the activations for the top α classes with the highest activation values.

4) Akar [8]: The work by Akar et al. [8] is a state-of-the-art open-set detector specifically for time series. We refer to this approach as *Akar*. The Akar method uses Dynamic Time Warping (DTW) to compute the similarity between a test set time series and the barycenters of known devices.

We use AUPRC (Area Under Precision-Recall Curve) as the evaluation metric [19] 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 hyper-parameter 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.

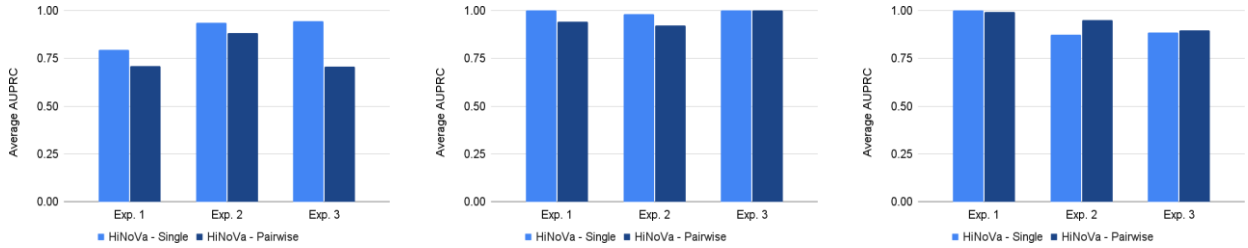
Specifically, we post-hoc tune the kernel size (2×256) and dropout rate (10%) in the CNN layer to achieve high accuracy in closed set classification using a grid search. Attaining good closed set accuracy has recently been shown to produce good open set detectors [5]. We also post-hoc tune the number 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). We report results with 25 bins in our experiments.

B. Experimental Results

Our performance evaluation is done using three different RF datasets: LoRa, Wireless-WiFi, and Wired-WiFi, as described in Sec. IV. Tables IIa, IIb and IIc show the average AUPRC values for the LoRa, Wireless-WiFi, and Wired-WiFi datasets respectively. HiNoVa consistently outperformed the other methods, achieving statistically significant results (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. Due to the extensive computational time of Akar, the algorithm did not complete within 24 hrs, making it infeasible to be used for this security use case.

Overall, the results suggest that HiNoVa is an effective detector of unknown devices using LoRa, Wireless-WiFi and Wired-Wifi protocols, outperforming other methods by a significant margin. The hidden state values correspond to a compact



(a) LoRa dataset (b) Wireless-WiFi dataset (c) Wired-WiFi dataset

Fig. 4: Average test AUPRCs for the single hidden node detector vs. the pairwise hidden node detector.

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 approaches and the OpenMax approach only 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.

C. 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). Figure 4 compares the performance of a single vs pairwise hidden node value detector. Figure 4 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.

VI. 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-WiFi, and Wired-WiFi 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.

VII. 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]," *IEEE Wireless Communications*, vol. 17, no. 5, pp. 63–70, 2010.

[2] A. Elmaghbub and B. Hamdaoui, "Lora device fingerprinting in the wild: Disclosing rf data-driven fingerprint sensitivity to deployment variability," *IEEE Access*, vol. 9, pp. 142 893–142 909, 2021.

[3] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1757–1772, 2013.

[4] 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.

[5] 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.

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

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

[8] 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.

[9] 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.

[10] 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.

[11] J. Gaskin, B. Hamdaoui, and W.-K. Wong, "Tweak: Towards portable deep learning models for domain-agnostic lora device authentication," *arXiv preprint arXiv:2209.00786*, 2022.

[12] 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.

[13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, jul 2009.

[14] S. Hanna, S. Karunaratne, and D. Cabric, "Open set wireless transmitter authorization: Deep learning approaches and dataset considerations," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 59–72, 2020.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[16] M. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. (1-2), pp. 81–89, 1938.

[17] 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.

[18] 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.

[19] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PloS one*, vol. 10, no. 3, p. e0118432, 2015.