Flow-Based Anomaly Intrusion Detection Systems using Recurrent Neural Networks

Hafiz Yasir Noor[1], Isaac Woungang[1], Glaucio H.S. Carvalho[2], Issa Traore[3], Dao Thanh Hai[4]
[1]Department of Computer Science, Toronto Metropolitan University, Toronto, ON, Canada.
[2]Department of Computer Science & Engineering, Brock University, St. Catharines, Canada
[3]Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada
[4]School of Science, Engineering & Tech, RMIT University, Vietnam
Emails: yasir.noor@torontomu.ca, iwoungan@torontomu.ca, gdecarvalho@brocku.ca,
itraore@ece.uvic.ca, hai.dao5@rmit.edu.vn

***Abstract*—As Internet of Things (IoT) networks continue to evolve, they face increasing security threats, and methods to achieve the security properties and requirements of these networks from the perspective of data, communication and IoT device security, are still on demand. This paper focuses on Recurrent neural network (RNN)-based methods, which provide intrusion detection systems (IDSs) with the capability of analyzing unseen and complex patterns in exchanges between IoT devices. Two flow-based optimized standalone RNN-based IDSs (called Uni-Hybrid and Bi-Hybrid RNN-based IDSs) are proposed to enhance the security of IoT networks. Through experiments using the IoTID20 dataset, the proposed models yield some marked improvements over a chosen benchmark model in terms of precision, accuracy, recall, and F1-score, highlighting the potential of RRN-based models in addressing intrusion detection in IoT networks.**

***Index Terms*—Internet of Things (IoT), Recurrent neural network (RNN), Uni-hybrid RNN, Bidirectional hybrid RNN, Intrusion detection system (IDS), Anomaly detection, Flow-based approach.**

I. Introduction

Internet of Things (IoT) is defined as a network of objects with embedded technologies and connectivity that facilitate the data exchanges among them for the provision of services. Such network poses serious security challenges due to the heterogeneous and distributed nature of its devices, making them more vulnerable and exploitable by attackers [2].

In an IoT environment, the IDS concept has been widely applied due to the advent of cloud computing, giving rise to the traditional IoT-oriented IDSs that have been designed to protect malicious activities on IoT devices and gateways. With the advent of edge computing as an extension of the cloud computing paradigm, the presence of edge nodes has opened new breaches that attackers could exploit to lunch attacks that traditional IoT-oriented IDSs cannot detect. This issue has prompted the need to design new strategies that a system can follow for detecting intrusions, namely signature-based IDSs and anomaly-based IDSs.

In a signature-based IDS, the signatures of known attacks are collected and contrasted against those of current activities, and an alarm is raised in case a matching is found. In anomaly-based IDSs, the nominal system behavior is built and compared with that of any system's computer host, and an alarm is raised if there is a significant deviation between those behaviors when the IDS is activated.

Anomaly-based IDSs can be classified into three main categories, namely statistics-based IDSs, knowledge-based IDSs, and machine learning (ML)-based IDSs. In statistics-based IDSs, events that occur in the system are monitored according to the system's probability distribution model during the learning process. Based on this model, alerts might be triggered by the IDS in case of low probability events. In knowledge-based IDSs, a profile of legitimate traffic signature is build based on a knowledge source representing such traffic, and any event that differs from this profile is categorized as anomalous. In ML-based IDSs, upon triggering the IDS, the system's normal behavior is learned during the ML training phase, and anomalous events that have occurred in the system are reported. For the latter category, many types of ML models have been proposed [1], which use various techniques to leverage the training strategy of data features while removing the need of hand-crafting the information for the IDS. Recent examples of such models are Recurrent Neural Networks (RNNs) and their instances such as the Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs), which provide the IDS with the additional capability of analyzing unseen and complex patterns in exchanges that occur between the hosts, which statistics-based or knowledge-based IDSs cannot do. This paper focuses on this later category of IDSs, by proposing two

979-8-3503-9281-4/25/$31.00 ©2025 IEEE

flow-based optimized standalone RNN IDSs (referred to as Uni-Hybrid and Bi-Hybrid RNN models) for enhancing the security of IoT networks.

Unlike packet level-based anomaly detection schemes, where the raw traffic is analyzed and malicious activities are identified by inspecting only few packets, flow-based anomaly detection schemes relies on a third-party component to generate the flow information from the network communication. For our proposed RNN-based IDSs, the IoTID20 dataset [3] is leveraged to extract the flow-level features that are used for optimizing the proposed Uni-Hybrid and Bi-Hybrid RNN models using the grid search algorithm [4]. These optimized models are then evaluated and compared against a benchmark model [1] in terms of precision, accuracy, recall, and F1-score.

The remainder of the paper is organized as follows. Section II discusses representative related works. Section III describes the proposed Uni-Hybrid RNN-based and Bi-Hybrid RNN-based IDSs. Section IV presents the methodologies. Section V is devoted to the performance evaluation of the proposed models. Section VI concludes the paper.

II. Related Works

In this section, representative works in the literature [1], [2], [5]- [9], are discussed.

In [1], Ullah et al. proposed a deep convolutional neural network (DCNN) model combined with feature engineering techniques, for intrusion detection purpose in IoT networks. Their model included two convolutional layers and three fully connected dense layers, and optimizer techniques such as Adam, AdaMax, and Nada were considered. Through simulations, their model achieved 99.91% accuracy, 99.87% precision, 99.37% recall, and 99.62% F1-score for binary classification, using the IoTID20 dataset [3] when trained for 50 epochs.

In [2], Malik et al. proposed a lightweight IoT botnet detection model using a one-class K-nearest neighbor classifier. Their approach was tested on multiple datasets, among which one yielded the best possible results, i.e. an accuracy of 99%, a precision of 99%, a recall of 100%, and a F1-score of 99%.

In [5], Protogerouet al. introduced a graph neural network (GNN)-based model that utilizes a distributed intelligence architecture across IoT devices, where agents are deployed to perform real-time anomaly detection using time series network data. Their model trained on the IoT-NI dataset for 100 epochs achieved an accuracy of 99% in detecting anomalies.

In [6], Abusit *et al.* proposed a deep learning-based anomaly detection scheme for IoT networks, in which useful features are selected based on a ML classifier to enhance the accuracy in detecting malicious IoT data. A denoising auto-encoder scheme is designed to filter the large-scale heterogeneous unlabeled data and retain the aforementioned useful features after a deep neural network model is pre-trained. Such auto-encoder scheme consists of two layers, i.e. the neutral layer which isolates the features that are deemed unnecessary and the decision layer, which does the retention of the useful features to be utilized by the classifier for distinguishing the benign IoT data from malicious ones. Experiments showed that the proposed model achieved promising results in terms of malicious IoT data detection accuracy.

In [7], Nie *et al.* proposed a multi-view learning architecture tailored for intrusion detection in IoT networks. Their approach used a representation of IoT traffic across multiple views (spatio-temporal, header field pattern, and payload semantic) to improve the detection accuracy across multiple tasks. Tested on the IoT-NI dataset, their model achieved 99.89% as accuracy, 97.09% as precision, 98.93% as recall, and 97.96% as F1-score for anomaly detection.

In [8], Gueriani *et al.* proposed an IDS model that combines the strength of LSTM and CNN deep learning models for classifying the IoT traffic as benign or malicious. This is achieved by exploiting the memory retention capability of Long Short-Term Memory (LSTM) and the feature extraction capability of Convolutional Neural Network (CNN) in recognizing patterns in data. Through simulations using the CICIoT2023 dataset, their model yields promising results in terms of malicious traffic detection capability, under predefined performance metrics.

In [9], Ding *et al.* proposed a Generative Adversarial Network (GAN)-model for intrusion detection in IoT networks, which consists of a multi-generator structure that simultaneously generate different attack types, and a classifier model that optimizes the discriminator and generator based on a classification loss parameter. A high-dimensional feature extractor is used to generate samples with less class overlap, and determine the cosine similarity between the original and generated samples. In such approach, the GAN model is optimized in the training phase, where the generator (resp. discriminator) is engaged in continuous optimization using a minimal game. The model is validated by experiments using three datasets, showing its superiority to a benchmark IDS model in terms of recall, precision, and F1-score.

Unlike the above works, this paper proposes novel flow-based Uni-Hybrid and Bi-Hybrid RNN models to enhance the anomaly detection in IoT networks.

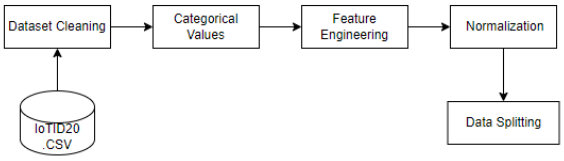

Fig. 1. Flow-based dataset pre-processing

```
Number of features with importance > 0.001: 63
Selected features (62): ['Dst_Port' 'Timestamp' 'Flow_ID' 'Src_Port' 'Flow_Pkts/s' 'ACK_Flag_Cnt'
 'Fwd_Pkt_Len_Std' 'Bwd_Pkts/s' 'Fwd_Pkt_Len_Max' 'Src_IP' 'Fwd_Pkts/s'
 'Flow_Duration' 'Init_Bwd_Win_Byts' 'Protocol' 'Flow_Byts/s'
 'Flow_IAT_Max' 'Dst_IP' 'Bwd_Header_Len' 'Flow_IAT_Mean' 'Pkt_Len_Min'
 'Pkt_Len_Max' 'Idle_Max' 'Pkt_Len_Std' 'Fwd_Pkt_Len_Min'
 'Fwd_Seg_Size_Avg' 'Fwd_Pkt_Len_Mean' 'Flow_IAT_Min' 'Bwd_Seg_Size_Avg'
 'Bwd_IAT_Tot' 'Idle_Min' 'Idle_Mean' 'Flow_IAT_Std' 'Idle_Std'
 'Pkt_Size_Avg' 'Pkt_Len_Mean' 'Bwd_Pkt_Len_Max' 'Bwd_Pkt_Len_Min'
 'Tot_Bwd_Pkts' 'Bwd_IAT_Mean' 'Down/Up_Ratio' 'Subflow_Bwd_Pkts'
 'Fwd_Header_Len' 'Pkt_Len_Var' 'Bwd_Pkt_Len_Mean' 'Bwd_IAT_Max'
 'TotLen_Bwd_Pkts' 'Tot_Fwd_Pkts' 'Subflow_Fwd_Pkts' 'Bwd_IAT_Std'
 'Fwd_IAT_Tot' 'Subflow_Bwd_Byts' 'Bwd_IAT_Min' 'Fwd_IAT_Min'
 'Fwd_IAT_Max' 'Fwd_Act_Data_Pkts' 'Fwd_IAT_Mean' 'SYN_Flag_Cnt'
 'TotLen_Fwd_Pkts' 'Subflow_Fwd_Byts' 'Bwd_Pkt_Len_Std' 'Bwd_PSH_Flags'
 'PSH_Flag_Cnt']
Shape of the dataset after feature selection: (625415, 62)
```

Fig. 2. Flow Based Selected Features.

III. Proposed Uni-Hybrid RNN- and Bi-Hybrid RNN-Based IDSs

We consider the model proposed in [1] as benchmark scheme, and we use the same IoTID20 dataset [3], and carry out a data pre-processing technique as detailed in the sequel. For the ML model design, the authors in [1] manually experimented with multiple combinations of hyoerparameters using the dataset and trained their model for 50 epochs. In contrast, we opted to use the grid search algorithm [4] to determine the optimal hyperparameters based on the dataset, and the `early stopping` technique [10] to enhance the training efficiency of our proposed models.

A. Flow-Based Dataset Preprocessing

The flow-based features are derived from the traffic flow rather than the individual packets within that flow. In this case, for anomalies to be detected, the flow needs to be completed. In this approach, not all extracted features are necessarily part of the standard TCP/IP headers. The steps of the dataset preprocessing (Fig. 1) using the IoTID20 dataset [3] are: (1) *IoTID20 dataset*: this .cvs file (inherited from the authors in [3]) contained 83 flow-based network features, features. It serves as input to the flow-based dataset preprocessing process; (2) *Dataset cleaning*: handling empty or undefined data instances is crucial to ensure that the data are ready for ML model training. To achieve this, we use an approach in which data records that contain missing values or undefined data instances are removed using `Pandas` Python libraries; (3) *Categorical values*: to convert the categorical features in the IoTID20 dataset [3] into numerical values, a label encoder method [10] is utilized; (4) *Feature engineering*: for the considered IoTID20 dataset [3], the Extra Trees classifier technique [10] is applied for feature selection. The importance of each feature is determined based on the output label, then the top 62 features in Fig. 2 are retained based on their sorted importance and information gain; (5) *Normalization*: the Min-Max normalization approach [10] is used; (6) *Data splitting*: the stratified method [10] is utilized to split the IoTID20 dataset [3] into homogeneous subsets, ensuring that each class is represented proportionally. The data is divided into 80% training and 20% test sets for each class as shown in Table I, noting that label 0 is assigned to normal packets and label 1 is assigned to anomalous packets.

TABLE I
Dataset Training and Testing Data Distribution

Class	Instances	Training Set	Testing Set
Malicious	585,342	468,274	117,068
Benign	40,073	32,058	8,015
Total	625,415	500,332	125,083

B. Optimal Hyperparameters Derivation

The grid search algorithm [4] is used to derive the optimal hyperparameters such as learning rate, batch size, choice of optimizer, and to determine the best network architecture, i.e. the ideal number of layers and neurons to maximize the performance of the studied models. It should be noted that increasing the number of hyperparameters leads to an increase in the number of model's configurations that should be evaluated. As a result, the grid search process had to explore more combinations of hyperparameters, which required high computational power and increased time consumption. The hyperparameters that were used and their respective options are given in Table II.

TABLE II
Hyperparameter Options Used in the Grid Search Algorithm.

Hyperparameter	Values
LSTM units	Layer 1: 256 units
	Layer 2: 128 units
	Layer 3: 64 units
Learning rate	0.001, 0.0001
Optimizers	Adam, Adamax, Nadam
Number of LSTM layers	1, 2, 3
Number of dense layers	1, 2
Number of epochs	3
Batch size	32, 64, 128, 256

Each combination of these hyperparameters is trained on the IoTID20 dataset [3], and the performance of each studied model is evaluated based on the accuracy, precision, recall, and AUC as performance metrics. The grid search encompasses a total of 144 unique combinations of hyperparameters, ensuring a thorough exploration of the potential configurations. The key steps in the grid search process (Fig. 3) are

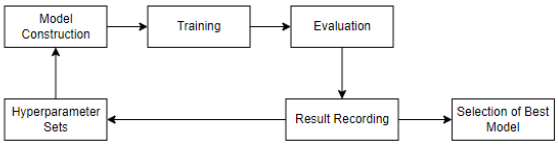

Fig. 3. Flowchart of the Grid Search Algorithm.

as follows: (a) *Given sets of parameters:* the process starts with the predefined sets of hyperparameters; (b) *Model construction:* for each hyperparameters' combination, a new model is constructed with the specified LSTM units, learning rate, optimizer, and layers; (c) *Training:* each model is trained for 3 epochs on the flow-based training set, using a predefined batch size. The training process involves optimizing the binary cross-entropy loss function, with accuracy, precision, recall, and AUC tracked at each epoch to assess the model's performance; (d) *Evaluation:* after training, the models are evaluated using the test dataset; (e) *Recording of results:* the performance metrics for each combination of hyperparameters are evaluated and recorded; and (f) Selection of best model: the performances of the studied models are compared based on the considered matrices given in Table III and the model that generated the optimal results is considered as the best one.

C. Flow-Based Optimized Hybrid Standalone RNN IDS Architecture

The grid search enabled the derivation of the optimal hyperparameters' configuration given in Table III.

TABLE III
FLOW-BASED OPTIMIZED HYBRID STANDALONE RNN IDS CONFIGURATION.

Hyperparameter	**Value**
`lstm_units_1`	256
`lstm_units_2`	128
`lstm_units_3`	64
`learning_rate`	0.001
`optimizer`	adam
`num_layers`	3
`num_dense_layers`	2
`epochs`	3
`batch_size`	256
Metrics	**Value**
`loss`	0.002983
`accuracy`	0.999161
`precision`	0.999547
`recall`	0.999556
`auc`	0.99979

The configuration in Table III resulted in outstanding performance metrics, with the model achieving a loss of 0.002983, accuracy of 99.91%, precision of 99.95%, recall of 99.95%, and an AUC of 0.99979.

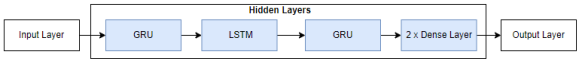

Fig. 4. Flow-based Optimized Standalone Uni-Hybrid RNNs IDS Architecture.

These results highlight the model's exceptional ability to classify the data accurately, with high precision and recall, while minimizing the false alarms.

The proposed Uni-Hybrid RNN model architevturre is depicted in Fig. 4. The model begins with an `Input layer` that accepts sequences with a shape defined by `(62, 62)`, where 62 represents both the number of `timesteps` and the number of features. This is followed by the `GRU, LSTM, and GRU Layers` consisting of 256, 128, and 64 units, respectively. To prevent overfitting, `Dropout layers` with a dropout rate of 0.2 are incorporated after each hidden layer. In addition, `Batch normalization layers` are applied after the first two GRU and LSTM layers. These layers normalize the activations of the previous layers and stabilize the training process. After the recurrent layers, the model includes two fully connected `Dense layers` with 128 and 64 units, respectively. These layers utilize the `ReLU` activation function, introducing non-linearity into the model. The `Dense output layer` comprises a single unit with a `sigmoid` activation function. This layer is designed for binary classification tasks, where it produces a probability score indicating the likelihood of the input sequence belonging to the malicious or benign class. For the proposed Bi-Hybrid RNN model, we adopted the architecture depicted in Fig. 4, but where the GRU and LSTM layers are replaced with their bidirectional counterparts.

D. Performance Evaluation

The considered performance metrics are:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall},} \tag{4}$$

where TP denotes True Positive, TN denotes True Negative, FN denotes False Negative and FP denotes False Positive.

TABLE IV
CLASSIFICATION OUTCOMES.

Actual	Predicted	Outcome	Description
1	1	TP	Anomaly detected
0	0	TN	Benign packet correctly classified
1	0	FN	Anomalous packet undetected
0	1	FP	Benign packet marked as anomalous

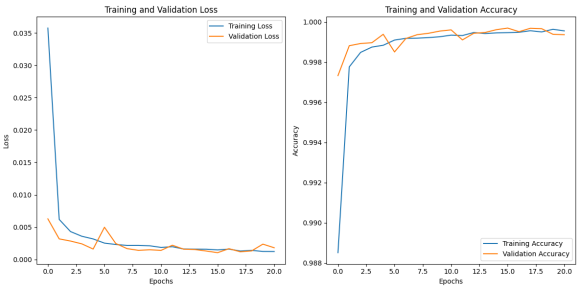

Fig. 5. Uni-Hybrid RNNs Standalone IDS Training and Validation Loss (left) and Accuracy (right)

E. Flow-Based Hybrid Standalone RNN IDSs Training and Results

For our proposed flow-based optimized standalone RNNs IDSs models training, we have utilized the *early stopping* regularization technique [10] during training to prevent overfitting and increase the models' generalization to unseen data (testing set). The key idea behind `early stopping` is to monitor the model's performance on a validation dataset during training and to end the training process once the performance no longer improves. As shown in Algorithm 1, the model training process consists of compiling the model, defining callbacks, training the model, and monitoring the training progress.

Fig. 5 illustrates the training process of our flow-based optimized Uni-Hybrid RNNs standalone IDS model. The graphs show the training and validation loss and accuracy. It can be observed that the `early stopping` was triggered at the 21th epoch because of no improvement in the validation accuracy. It has restored the model's weights from the 16th epoch, the point at which the model achieved its best performance. The training and validation accuracy reached a peak at 99.9% by the 15th epoch.

Fig. 6 depicts the training process of our Bi-Hybrid standalone IDS model. It is observed that the `early stopping` was triggered at the 27th epoch after 5 consecutive epochs of no improvement in the validation accuracy. Consequently, the weights of the model were restored to those from the 22nd epoch where the best performance was achieved. The training loss

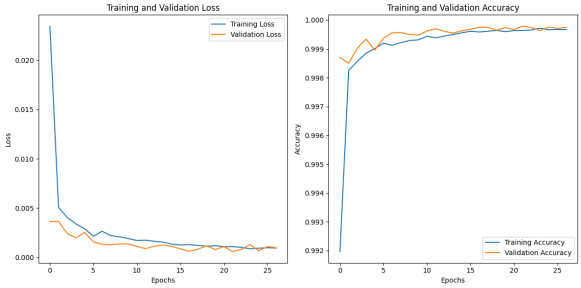

Fig. 6. Bi-Hybrid RNNs Standalone IDS Training and Validation (left) and Accuracy (right)

decreased sharply, reaching near-zero levels by the 10th epoch, and the validation loss remained low with minor fluctuations around 0.002. The training and validation accuracy increased to over 99.9% and was stabilized by the 10th epoch.

Algorithm 1 Training of Models

Step 1: Compile the model:
  a) Optimizer: Adam with learning rate of 0.001
  b) Loss Function: Binary Crossentropy
  c) Metrics: accuracy, precision, recall, AUC
Step 2: Define Callbacks:
  a) *Early Stopping:*
    i) Monitor `val_accuracy`
    ii) Patience: 5 epochs
    iii) Restore best weights
  b) *Model Checkpointing*:
    i) Monitor `val_accuracy`
    ii) Save best model only
Step 3: Train the model:
  a) Use the `fit` method with `epochs=50`, `batch_size=256`, and `validation_split=0.2`
  b) Apply callbacks during the training phase
Step 4: Monitor the training:
  a) Track the metrics for each epoch
  b) Apply early stopping if no improvement for 5 epochs
  c) Save the best model if checkpoint criteria are met

Fig. 7 depicts the confusion matrices for our proposed flow-based models. The Uni-Hybrid model correctly identifies 116,993 anomalies (TP) and 7,995 benign packets (TN), but failed to detect 16 anomalies (FN) and incorrectly flagged 18 benign packets as anomalies (FP). In contrast, the Bi-Hybrid model shows a slightly better anomaly detection, with 117,001 true positives and 8 reduced false negatives, but at the cost of increasing the false positives to 30, with 7,983 benign packets correctly classified.

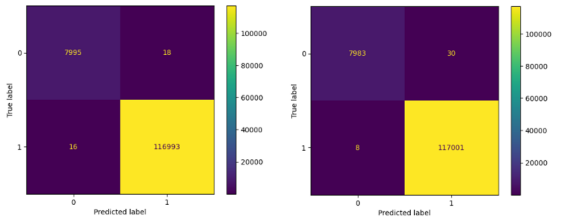

Fig. 7. Uni-Hybrid (left) and Bi-Hybrid (right) Confusion Matrices

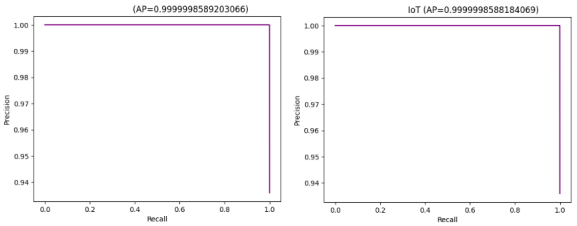

Fig. 8. Uni-Hybrid RNN/Bi-Hybrid RNN Average Precision (AP)

Fig. 8 shows the precision-recall curves for the Uni-Hybrid and Bi-Hybrid models, respectively. Both models demonstrate a high performance, with near-perfect Average Precision (AP) scores of about 0.9999999. The precision-recall curves for both models are almost identical, representing how they maintain a precision of nearly 1.0 across all recall levels.

TABLE V
EVALUATION OF FLOW-BASED STANDALONE RNNS MODELS.

Model	Accuracy	Precision	Recall	F1-Score
Ullah et al. [1]	99.91%	99.87%	99.38%	99.62%
Uni-Hybrid IDS	99.973%	99.985%	99.986%	99.998%
Bi-Hybrid IDS	99.970%	99.974%	99.993%	99.998%

The ROC curves for the Uni-Hybrid and Bi-Hybrid RNNs IDS models in Fig. 9 demonstrate their classification performance. The Uni-Hybrid IDS model achieves the highest AUC of 0.99999794, indicating a near-perfect ability to distinguish between true positives and false positives. On the other hand, the Bi-Hybrid IDS model followed closely with an AUC of 0.99999793, also reflecting an exceptional performance. From Table V, it is found that compared to the benchmark model [1], our proposed Uni-Hybrid and Bi-Hybrid standalone IDS models yield significant improvements across all metrics.

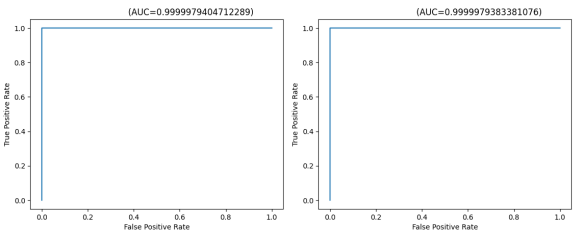

Fig. 9. AUC for Uni-Hybrid RNN (left) and AUC for Bi-Hybrid RNN (right).

IV. CONCLUSION

We have developed two flow-based optimized hybrid standalone IDSs (i.e. Uni-Hybrid and Bi-Hybrid), then trained and evaluated them on the IoTID20 [3] dataset. These models demonstrated higher performance than the benchmark model across all metrics. While the Bi-Hybrid IDS achieved the highest recall at 99.993%, our Uni-Hybrid IDS delivered a well-balanced performance, achieving 0.063%, 0.115%, 0.606%, and 0.378% higher accuracy, precision, recall, and F1-score, respectively. It is worth noting that the benchmark model [1] was trained for 50 epochs, while our Uni-Hybrid model needed only 16 epochs and our Bi-Hybrid model needed only 22 epochs.

REFERENCES

[1] S. Ullah, J. Ahmad, M.A. Khan, E.H. Alkhammash, M. Hadjouni, Y.Y. Ghadi, F. Saeed, N. Pitropakis, 'A new intrusion detection system for the internet of things via deep convolutional neural network and feature engineering", Sensors, vol. 22, no. 10, p. 3607, 2022. 3607.

[2] K. Malik, F. Rehman, T. Maqsood, S. Mustafa, O. Khalid, and A. Akhunzad, "Lightweight Internet of Things Botnet Detection Using One-Class Classification", Sensors, vol. 22, no. 10, pp. 36-46, 2022.

[3] I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous activity detection in iot networks," in Proc. of the Canadian Conference on Artificial Intelligence, Ottawa, Canada, 2020, pp. 508–520.

[4] J-M. Dufour, J. Neves, "Finite-sample inference and nonstandard asymptotics with Monte Carlo tests and R", Chapter 1 in Handbook of Statistics, Elsevier, by Hrishikesh D. Vinod, C.R. Rao (Ed.), vol. 41, 2019, pp. 3-31.

[5] A. Protogerou, E. V. Kopsacheilis, A. Mpatziakas, K. Papachristou, T. I. Theodorou, S. Papadopoulos, A. Drosou, D. Tzovaras, "Time Series Network Data Enabling Distributed Intelligence—A Holistic IoT Security Platform Solution", *Electronics*, vol. 11, pp. 5-29, 2022.

[6] A. Abusitta, G.H. de Carvalho, O.A. Wahab, T. Halabi, B.C. Fung,, and S. A. Mamoori, "Deep learning-enabled anomaly detection for iot systems", Internet of Things, 2023, vol. 21, pp. 100-656.

[7] F. Nie, W. Liu, G. Liu, B. Gao, "M2VT-IDS: A multi-task multi-view learning architecture for designing IoT intrusion detection system," Internet of Things, vol. 25, 2024, p. 101102, 2024.

[8] A. Gueriani, H. Kheddar, A. C. Mazari, "Enhancing IoT Security with CNN and LSTM-Based Intrusion Detection Systems," Proc. of the IEEE Intl. Conference on Artificial Intelligence, 2024, pp. 1-8.

[9] H. Ding, Y. Sun, N. Huang, Z. Shen, X. Cui, "Tmg-gan: Generative adversarial networks-based imbalanced learning for network intrusion detection", IEEE Trans. on Information Forensics and Security, vol. 19, pp. 1156–1167, 2024.

[10] scikit-learn, https://scikit-learn.org/stable/ (Last visited March 25, 2025)