I. J. Computer Network and Information Security, 2026, 3, 44-62
Published Online on June 8, 2026 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2026.03.03

A Hybrid MAML-Reptile Based Few-shot Learning Approach for Securing Fog-IoT Networks against Maleficent Behaviors

Surya Pavan Kumar Gudla
Department of Computer Science and Engineering, NCR- PMEC Berhampur, Faculty of Engineering, Biju Patnaik University of Technology (BPUT), Rourkela, 769015, India
E-mail: pavan1980.mca@gmail.com
ORCID iD: https://orcid.org/0000-0002-6871-5841

Sourav Kumar Bhoi*
Department of Computer Science and Engineering, Parala Maharaja Engineering College (Govt.), Berhampur, 761003, India
E-mail: sourav.cse@pmec.ac.in
ORCID iD: https://orcid.org/0000-0002-5173-3453
*Corresponding author

Kshira Sagar Sahoo
Department of Computer Science and Engineering, SRM University, Amaravati, 522502, India
E-mail: kshirasagar12@gmail.com
ORCID iD: https://orcid.org/0000-0002-6435-5738

GNV Rajareddy
Department of Chemical and Biological Engineering, University of Saskatchewan, Saskatoon, Canada S7H 5A2
E-mail: cre138@usask.ca
ORCID iD: https://orcid.org/0000-0003-2394-2693

Received: March 7, 2025; Revised: September 15, 2025; Accepted: December 18, 2025; Published: June 8, 2026

Abstract: Security in Fog based IoT networks has a major problem where some IoT devices may be compromised due to attacks. This creates vulnerability to the sensitive data flow between Fog and IoT devices. Also, due to heterogeneity in network structure, the probability of attacks in the network is more. To analyze this dynamic and complex structure of fog networks, few-shot learning is a good solution to learn patterns more accurately and identify the maleficent or normal behavior of a device. In deep learning, few-shot learning is a technique that uses a small amount of labeled data for data processing that is more efficient than the traditional deep learning models. In this work, a hybrid MAML (Model Agnostic Meta Learning)-Reptile based few shot learning approach is proposed that secures the fog-based IoT infrastructure from a variety of attacks by detecting the attacks more accurately. The few shot models considered are Prototypical Networks, Matching Networks, MAML, and Reptile for selection of best model to be run at the fog node for attack detection. The fog node uses the best model to detect the attacks and broadcasts the local behavior list (LBL) to cloud and other fog nodes to generate the global behavior list (GBL) for sharing more attack information with the IoT device layer. Here, we consider standard datasets such as UNSW-NB-15, NSL-KDD, and CICIDS-2017 for performing implementations. The performance of the models is analyzed using accuracy, recall, f1-score, precision, inference time, training time, AUC-ROC, cost, energy consumption, and processing latency. From the results, it is observed that the proposed MAML-Reptile hybrid model performs better than other standard models in detecting maleficent behaviors more accurately.

Index Terms: Security, Fog-IoT Network, Few Shot Learning, MAML-Reptile, GBL, LBL.

This work is open access and licensed under the Creative Commons CC BY 4.0 License.

1. Introduction

Fog computing is an architecture that is decentralized and it extends the services of cloud computing nearer to the edge of the network [1-6]. In a three-layered architecture, fog layer is at the intermediate layer between the Internet of Things (IoT) and cloud data centers. In the device layer, many devices such as sensors, smart devices, industrial machines, autonomous vehicles, etc. communicate with the fog layer through the edge layer for receiving services. Earlier, before this technology, the devices send their requests to execute their tasks or applications in the cloud, that may take larger service time and increase the latency. Also, the bandwidth requirement is also high for the devices. This architecture reduces the network burden over the cloud by reducing latency, service time, and enabling quicker real-time decision making, etc.

Despite the many advantages of fog computing due to de-centralization and heterogeneity, it faces major security chal lenges [7-11]. The major security challenges in fog computing are data privacy and security, authentication and access control, Distributed Denial of Service attacks (DDoS), Spoofing attacks, data manipulation and tampering, intrusion attacks and malware infections, etc. As the threats are increasing severely, a system is needed that can monitor the attacks in real time. As many times sensitive applications run on Fog-IoT networks, immediate detection of attacks is highly needed. These systems can be deployed at the fog nodes for locally identifying the threats. The traditional security systems are inefficient due to latency and bandwidth requirement. Also, this attack detection system at the fog is scalable and adaptable because the IoT devices may dynamically join and leave the network. Also, the number of IoT devices is increasing and the network flow patterns are also diversified, where traditional machine learning based approaches lack in performance of detection of attacks.

Traditional ML based approach needs a large amount of labeled data for effective detection of cyber-attacks. Also, the attacks are changing frequently and also new attacks are generated and maintaining such large amount of labeled data is difficult. Few-shot learning is a meta-learning technique that enables the few-shot models to learn quickly from few labeled examples, and makes it more effective in these environments to quickly detect threats [12-14]. By applying a few-shot learning approach, frequent retraining is reduced with high detection accuracy. There are many few-shot learning models, such as prototypical networks, matching networks, model agnostic meta learning (MAML), Reptile, etc.

MAML and Reptile have both shown potential in meta-learning. MAML has rapid adaptive nature with computational overhead, Reptile gives computational efficiency but risks underfitting in different maleficent behaviors. This makes it difficult to apply each meta-learner individually. The limitations of MAML are counterbalanced by the strengths of Reptile, and vice versa, making them suitable for Fog-IoT security. Theoretically, our proposed hybrid MAML-Reptile architecture aims to combine the adaptability nature of MAML with computational efficiency of Reptile. This combination is critical for the real-time identification of maleficent behaviour.

From the above study, the major contributions in this work are stated as follows.

- In this work, a hybrid MAML-Reptile based few shot learning approach is proposed for attack detection in Fog-IoT networks.
- The hybrid MAML-Reptile based few shot learning is compared with Prototypical Networks, Matching Networks, MAML, and Reptile for selection of best model for deployment at the fog node in Fog-IoT networks. The standard datasets considered for this are UNSW-NB-15, NSL-KDD, and CICIDS-2017.
- The attacks detected using the best model at fog node are broadcast in the network using local behavior list (LBL) and global behavior list (GBL).
- From the results, it is observed that hybrid MAML-Reptile based few shot learning model performs better than other standard few-shot learning models in detecting the maleficent behaviors in Fog-IoT networks.

The next sections are discussed as follows. Section 2 presents the recent related works on few-shot learning for attack detection. Section 3 presents the methodology that describes the system model, attack model, few shot learning, datasets, hybrid model, and working of the model. Section 4 presents the results and discussion. Section 5 presents the conclusion and future scope of the work.

2. Related Works

In this section, we discuss related work on attack detection in fog computing using a few-shot learning approach. Gamal et al. [15] proposed a new deep learning based few-shot algorithm called CNN-IDS. This can identify zero-day attacks on the network. Islam et al. [16] studied the usage of lightweight IDS for detecting attacks in the IoT environment. Samy et al. [17] studied the performance of 6 deep learning models to detect cyber attacks. Demirpolat et al. [18] proposed an intelligent SDN based security architecture in IoT systems using the few-shot learning approach. The prototypical network is used as a classifier for this purpose. Thein et al. [19] proposed a method to detect attacks using a prototypical graph neural network. Yang et al. [20] proposed a method known as FS-IDS to detect attacks using the few-shot learning approach. Hosseini et al. [21] proposed a few-shot class incremental learning for attack detection

in IoT networks. Diro et al. [22] detected the distributed cyber attacks using LSTM network for fog-to-things communication. Dina et al. [23] present a few-shot and self-supervised framework for detection of attacks in IoT systems. Tian et al. [24] integrate Bi-LSTM and multi-head self-attention for detecting attacks in fog enabled IoT systems.

From the above study, it is seen that most approaches used single deep learning models or traditional few-shot learning approach. Also, these works have not explored meta learning optimization such as the use of Reptile and MAML. Also, many research works do not compare the proposed model with the standard few shot learning models. This motivates us to design a hybrid MAML-Reptile based few-shot learning approach for attack detection in Fog-IoT networks.

3. Methodology

3.1. System Model

The system model consists of a three-layer architecture as shown in Fig. 1. The bottom layer is the device layer where the IoT devices exist. The middle layer is the fog layer where the fog nodes exist. The top layer is the cloud layer where the cloud node exists.

IoT Layer: IoT layer consists of IoT devices where many IoT devices exist to perform any task or operations. The main building blocks of these IoT devices are memory, processor, power supply and transceiver. These devices have a limited memory size, limited computational power to process tasks, a limited power supply, and a transceiver to send and receive data. The IoT devices communicate using wireless communication over 5G/6G networks. They also communicate with fog nodes via base station which are in the communication range for processing their queries or executing applications. Also, they can communicate with the cloud node through fog node for getting other services.

Fog Layer: Fog layer consists of fog nodes that are connected to each other. Each fog node can provide services to IoT devices that are in the communication range or near proximity. The fog node has high computational capability, high power supply, and can provide services at the edge of the network by reducing the latency and burden on the cloud infrastructure. The fog node can communicate with other fog nodes, cloud node and IoT devices using wireless communication (5G/6G).

Cloud Layer: Cloud layer consists of a cloud node that is basically used to provide services such as infrastructure as a service (IaaS), storage as a service (SaaS), network as a service (NaaS), and computing as a service (CaaS). Cloud node has huge data centers to provide such services. Cloud node needs a high bandwidth internet facility to communicate with fog nodes.

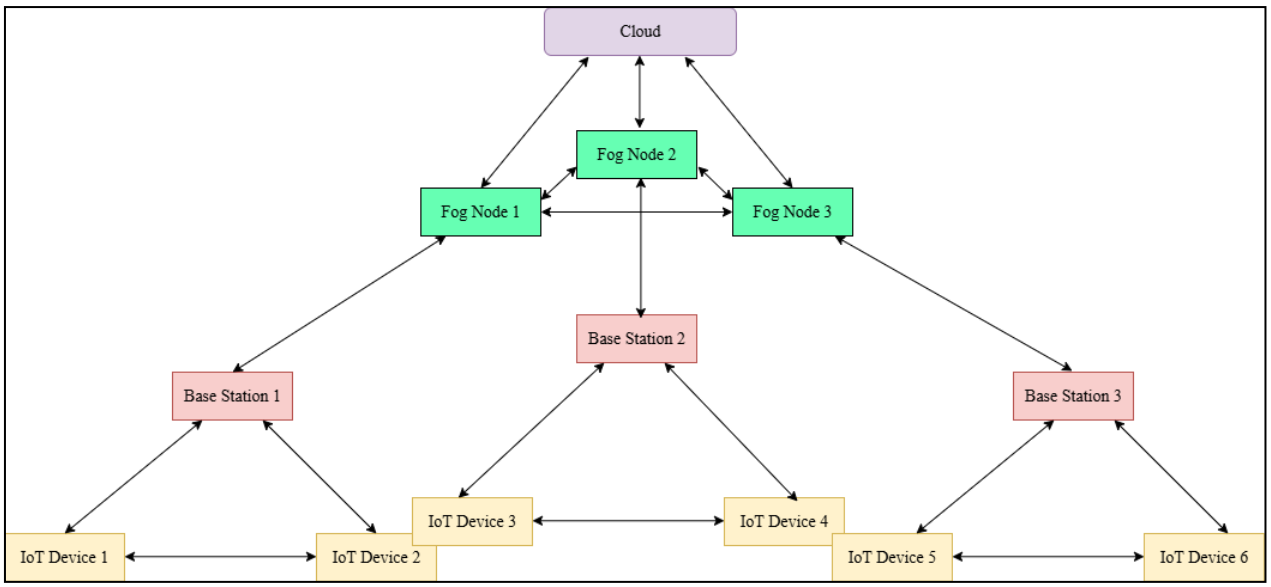


Fig. 1. Three-layer cloud-fog-based IoT network architecture.

3.2. Attack Model

The attack model discusses the attacks possible in the Fog-IoT networks. The attacker attacks the IoT devices in many ways and compromises the IoT devices for performing many attacks at the device layer or fog layer. The main attacks considered in this work include DoS, DDoS, Bruteforce, Botnet, Infiltration, Web attacks, Port scanning. These attacks are taken from CICIDS 2017 dataset. From UNSW-NB15, the attacks considered are Fuzzers, Analysis, Backdoor, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. From NSL-KDD, the attacks considered

are DoS, Probe, U2R, and R2L.

3.3. Few-Shot Learning Models

A. Prototypical Networks

Prototypical networks are the networks that are a metric based meta learning approach designed for few-shot learning [25]. It learns by mapping the examples to the feature space. Here, each class is represented by a prototype vector. In this manner, the new examples are classified by calculating the distance to these prototype vectors. The process for prototypical network consists of mainly three steps. Those are discussed as follows:

Step 1: Finding the class prototype by mean representation of each class. It is done by considering a support set S by taking a few labeled examples per class. A class C is represented by a prototype that is called as the mean of S. It is shown as follows.

$$\mathbf{m}_k = \frac{1}{|S_k|} \sum_{\mathbf{x}_i \in S_k} T_\theta(\mathbf{x}_i) \tag{1}$$

where, S_k is the set of labeled support examples of class k, $f_\theta(\mathbf{x}_i)$ is the feature embedding of input X_i, and c_k is the prototype of that class.

Step 2: Finding the distance between the new example and the class prototype. It is done by computing the new example X by finding the embedding as follows.

$$Z = f_\theta(\mathbf{x}) \tag{2}$$

Then the distance between z and c_k is calculated using Euclidean distance:

$$\delta(\mathbf{v}, \mathbf{m}_k) = \| \mathbf{v} - \mathbf{m}_k \|^2 \tag{3}$$

Step 3: Assigning probability to each class using the softmax function. The prototypes which are closer get high probability and farther prototypes get lower probability.

$$P(y = k \mid \mathbf{x}) = \frac{\exp(-\delta(\mathbf{v}, \mathbf{m}_k))}{\sum_j \exp(-\delta(\mathbf{v}, \mathbf{m}_j))} \tag{4}$$

B. Matching Networks

Matching networks are networks that are a metric based meta learning approach designed for few-shot learning [26]. The step by step process is discussed as follows:

Step 1: Generating feature representation for query and support data.

$$\mathbf{h}_i = f_\theta(\mathbf{x}_i), \quad \forall (\mathbf{x}_i, y_i) \in S \tag{5}$$

$$\mathbf{h}_q = f_\theta(\mathbf{x}_q) \tag{6}$$

Step 2: Measuring the relationship between query and support instances.

$$D(\mathbf{x}_q, \mathbf{x}_i) = \| f_\theta(\mathbf{x}_q) - f_\theta(\mathbf{x}_i) \|^2 \tag{7}$$

Step 3: Finding the influence weights in support set.

$$\alpha_i = \frac{\exp(-D(\mathbf{x}_q, \mathbf{x}_i))}{\sum_j \exp(-D(\mathbf{x}_q, \mathbf{x}_j))} \tag{8}$$

Step 4: Finding the final class and assigning the probability.

$$P(y_q = k \mid \mathbf{x}_q) = \sum_{i: y_i = k} \alpha_i \tag{9}$$

C. MAML

Model-Agnostic Meta Learning (MAML) is a gradient based meta learning model in few shots learning. Using very few examples, it quickly adapts to new tasks [27]. MAML mainly focuses on learning a model how to learn, by fine tuning itself with minimum number of updates.

Step 1: Defining the task specific training process.

$$T_i \sim P(T) \tag{10}$$

Step 2: Setting the learning parameters with $\mathbf{\Theta}_0$.

Step 3: Each task has a training and testing set that is done for task specific parameter updates.

$$\mathbf{\Theta}_{i'} = \mathbf{\Theta}_0 - \lambda \nabla_{\mathbf{\Theta}_0} L_{T_i}(M_{\mathbf{\Theta}_0}) \tag{11}$$

Step 4: Fine tuning is performed for meta optimization.

$$L_{meta} = \sum_i L_{T_i}(M_{\mathbf{\Theta}_{i'}}) \tag{12}$$

Step 5: The model parameters are updated across all the tasks.

$$\mathbf{\Theta}_0 \leftarrow \mathbf{\Theta}_0 - \eta \nabla_{\mathbf{\Theta}_0} L_{\text{meta}} \tag{13}$$

D. Reptile

Reptile is a gradient based meta learning approach that quickly adapts to new tasks and requires minimal fine tuning [28]. This model is similar to MAML but it is simple and it also does not need second order gradients. The step by step process is discussed as follows.

Step 1: Defining the setup of the few shots learning and selecting a sample task randomly.

$$T_i \sim P(T) \tag{14}$$

Step 2: Setting model parameters using $\mathbf{\Phi}_0$.

Step 3: Updating inner loop by task specific training.

$$\mathbf{\Phi}_{i'} = \mathbf{\Phi}_0 - \gamma \nabla_{\mathbf{\Phi}_0} J_{T_i}(F_{\mathbf{\Phi}_0}) \tag{15}$$

Step 4: Updating the outer loop across multiple tasks.

$$\mathbf{\Phi}_0 \leftarrow \mathbf{\Phi}_0 + \mu(\mathbf{\Phi}_{i'} - \mathbf{\Phi}_0) \tag{16}$$

Step 5: Meta update over multiple tasks.

$$\mathbf{\Phi}_0 \leftarrow \mathbf{\Phi}_0 + \mu \sum_i (\mathbf{\Phi}_{i'} - \mathbf{\Phi}_0) \tag{17}$$

where, $\mathbf{\Phi}$ is the model parameter, $F_{\mathbf{\Phi}}$ represent model, and J is the loss function.

3.4. Datasets

In this work, three standard datasets are considered, namely UNSW-NB-15, NSL-KDD, and CICIDS-2017. These datasets are described as follows.

UNSW-NB-15: This dataset is generated by ACSS and contains modern types of attacks [29]. It has a total of 2.54M records, with 49 features including the label. It has attack traffic of 2,218,761 and normal traffic of 321,283. It mainly consists of flow based features such as source, destination, protocol, state, duration, service, etc. It has content based features showing payload size, bytes, number of packets. It has time based features such as flow duration, inter packet arrival time. It has also statistical features such as mean, variance, etc. There are 9 types of attacks such as Fuzzers, Analysis, Backdoor, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms.

NSL-KDD: It has 148K records where the training set has 125,973 records and the testing set has 22,544 records [30]. The number of features including label is 42. It has basic features such as source bytes, destination bytes, protocol,

duration, etc. It has content based features such as failed login attempts, root access attempts, etc. Also, it has traffic based features such as percentage of connections to same host. The attacks in this dataset are DoS, Probe, R2L, and U2R.

CICIDS-2017: This dataset is developed by CIC and consists of real network traffic [31]. The total number of records is 2.8M and the number of features is more than 80. It has flow-based features such as flow duration, total packets, total bytes, etc. It has header-based features such as source and destination ports, protocol, etc. It also has behavioral features such as packet length variance, flow inter-arrival time, etc. The attacks in this dataset are Bruteforce, DoS, DDoS, Botnet, Infiltration, Web attacks, and Port scanning.

3.5. Hybrid MAML-Reptile Based Few Shots Learning Approach

In this work, two important few-shot learning algorithms, MAML and Reptile are hybridized. MAML is selected because it has a fast adaptation capability by optimizing a model with minimal fine-tuning of gradient updates. Reptile is selected because it is a simple meta-learning model and it moves the parameters towards average task specific solution. Also, it does not compute second order gradients. These two are very useful for attack classification, when new attack patterns come and it requires quick adaptation with less labeled data. The steps for the hybrid MAML-Reptile approach are discussed as follows:

Step 1: Randomly select a task sample from the task distribution.

$$T_i \sim P(T) \tag{18}$$

Step 2: Initializing the model parameters $\boldsymbol{\Psi}_0$.

Step 3: Updating the inner loop training for gradient updates for each task.

$$\boldsymbol{\Psi}_{i'} = \boldsymbol{\Psi}_0 - \lambda \nabla_{\boldsymbol{\Psi}_0} J_{T_i}(C_{\boldsymbol{\Psi}_0}) \tag{19}$$

Step 4: Reptile based updation by moving towards the adapted parameters.

$$\boldsymbol{\Psi}_0 \leftarrow \boldsymbol{\Psi}_0 + \rho(\boldsymbol{\Psi}_{i'} - \boldsymbol{\Psi}_0) \tag{20}$$

Step 5: Hybrid Meta-Update by combining MAML and Reptile.

$$\boldsymbol{\Psi}_0 \leftarrow \boldsymbol{\Psi}_0 - \tau \nabla_{\boldsymbol{\Psi}_0} \sum_i J_{T_i}(G_{\boldsymbol{\Psi}_i}) + \zeta(\boldsymbol{\Psi}_{i'} - \boldsymbol{\Psi}_0) \tag{21}$$

where, $G_{\boldsymbol{\Psi}}$ is the model and J is the loss function. In the above equations, τ and ζ are constants that regulate the MAML-style gradient, and ζ controls the interpolation strength of Reptile. The balance in choosing the values of τ and ζ is maintained to achieve a trade-off between accuracy and computational efficiency. The higher the τ value, the higher is the accuracy, but it increases cost. The higher the ζ value, the faster the convergence and the more stable the updates, but the lower the efficiency in heterogeneous tasks.

3.6. Methodology for Attack Detection in Fog-IoT Networks

The methodology for attack detection consists of two phases, the first is System Setup Phase and the second is Attack Detection in Fog-IoT Networks.

A. System Setup Phase

In this phase, the system is set as per the system architecture shown in Fig. 1. The IoT devices can communicate with each other or with the fog nodes using wireless communication. The fog node communicates with the cloud node. Fog nodes are trained and tested to detect attacks using a few-shot learning model along with the proposed MAML-Reptile based few shots learning model. The best model is selected and implemented at the fog nodes for attack detection. Here, the proposed hybrid model is compared with Prototypical Networks, Matching Networks, MAML, and Reptile. The standard datasets used for this training and testing are UNSW-NB15, NSL-KDD, and CICIDS-2017. The algorithm for this setup is shown in Algorithm 1 and Fig. 2.

Algorithm 1. Adaptive Few-Shot Learning Model Selection for Fog Nodes

Require: IoT devices, Fog Nodes, Cloud Node, Datasets (UNSW-NB15, NSL-KDD, CICIDS-2017)
Ensure: Best Few-Shot Model Deployment at Fog Nodes
1. **Begin**
   Phase 1: System Setup

2. Initialize a three-layer architecture: IoT devices, Fog Nodes, Cloud Node
3. Establish wireless communication between IoT devices and Fog Nodes
4. Enable communication between Fog Nodes and Cloud Node
   Phase 2: Data Collection and Preprocessing
5. Collect attack data from IoT devices
5. 6. Preprocess data (normalize, remove duplicates, extract features)
6. Split data into training and testing sets (using UNSW-NB15, NSL-KDD, CICIDS-2017)
   Phase 3: Few-Shot Learning Model Training
7. **for** each model in {Prototypical Networks, Matching Networks, MAML, Reptile, MAML-Reptile Hybrid} **do**
8. Train model on training set
   Phase 4: Model Evaluation and Selection
9. **for** each trained model **do**
10. Evaluate model on testing data
11. Compute performance metrics: accuracy, precision, recall, F1-score
12. Select the model with the highest accuracy as the best model
12. **13. Phase 5: Deployment at Fog Nodes**
13. Deploy the best-selected model at the Fog Nodes
14. Monitor real-time attack detection performance
15. **if** model performance degrades **then**
16. Adapt model based on new attack patterns
17. **END**

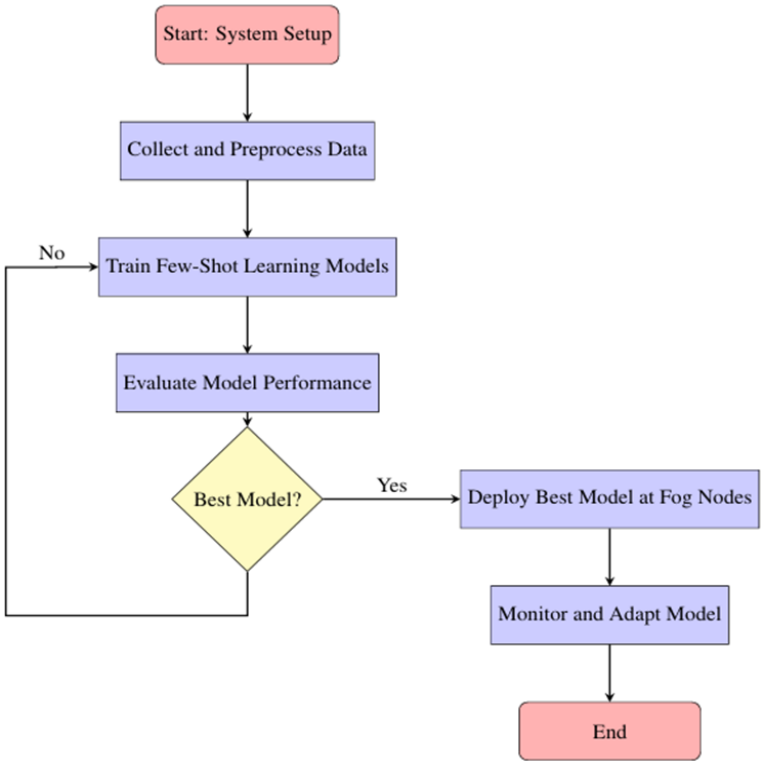


Fig. 2. Adaptive few-shot learning model selection for fog nodes.

B. Methodology of Attack Detection in Fog-IoT Networks

In the second phase, we deployed the best few-shot learning model at the fog nodes. The process flow diagram of the methodology is shown in Fig. 3. Now, the fog nodes are trained to detect attacks. Firstly, the IoT devices can communicate with each other to perform tasks such as execute applications, data forwarding, downloading/uploading, etc. After communication, the network flow of these communication transactions is sent to the fog node for attack detection. It is sent via the base station to the fog node. After receiving the new network flows from different IoT devices in its communication range, it uses the best few-shot learning model to classify the network flow pattern as attack or normal. The attack and normal list generated by the fog node for n network flows is the local behavior list (LBL). The LBL consists of IoT <device ID, behavior (attack / normal), timestamp>. This LBL is now broadcast to the cloud node C and to the connected fog nodes (FNs).

The LBL is broadcast to the cloud for data storage and update of the behavior of all IoT devices in the network.

Also, the LBL is broadcast to the connected FNs because the list needs to be shared with other IoT devices that are in the communication range of these fog nodes to inform them about the normal or attacker list. So, that in the future, before starting the communication, an IoT device will check whether another IoT device is an attacker or normal. When a fog node receives LBL from another fog node it merges all fog LBLs to form a global behavior list (GBL). This GBL will now be broadcast to the IoT devices in the communication range via base station. The GBL list is stored in the local database of an IoT device after receiving the GBL. This GBL is updated again and again when a new GBL is received by the IoT device. The algorithm and flowchart are mentioned in Algorithm 2.

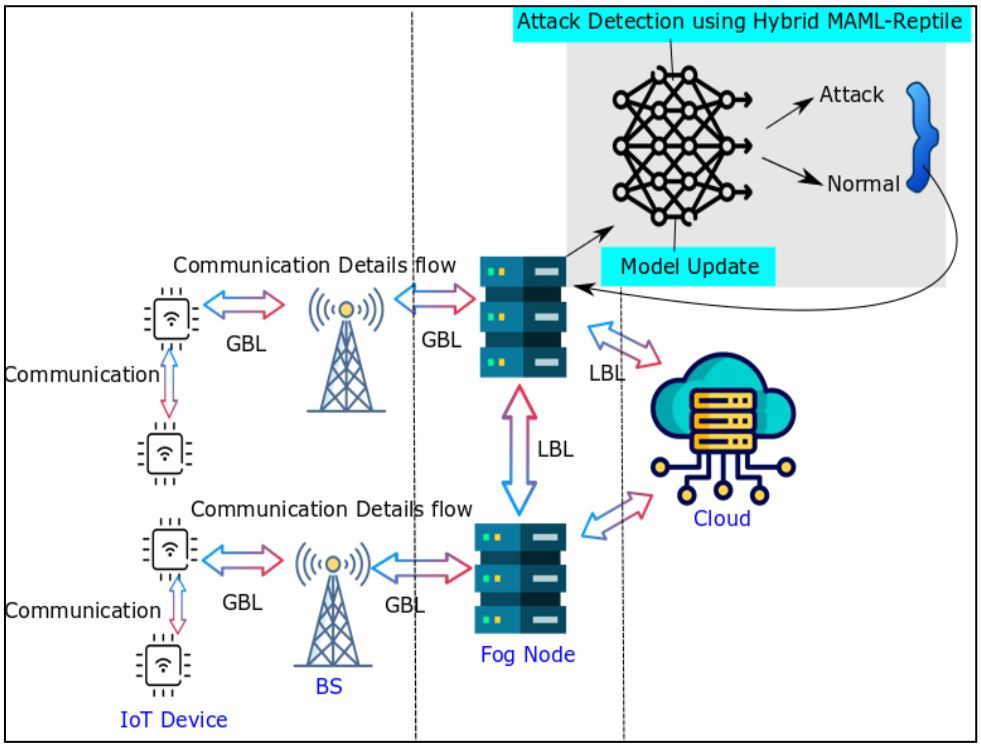


Fig. 3. Attack detection and behavior list management in Fog-IoT networks.

Algorithm 2. Attack Detection and Behavior List Management in Fog-IoT Networks

Require: IoT devices, Fog Nodes FNs, Cloud Node (C), Best Few-Shot Learning Model
Ensure: Attack Detection and Behavior List Distribution

1. **Begin**
   Phase 1: IoT Device Communication
2. **for** each IoT device D_i in the network **do**
3. Communicate with other IoT devices (data forwarding, downloading, executing applications)
   Phase 2: Network Flow Transmission to Fog Nodes
4. **for** each communication transaction T_j **do**
5. Send network flow data to the nearest Fog Node (FN_k) via base station
   Phase 3: Attack Detection at Fog Nodes
6. **for** each received network flow N_l at FN_k **do**
7. Classify N_l using the best few-shot learning model
8. Determine behavior: **attack** or **normal**
   Phase 4: Local Behavior List (LBL) Generation
9. **for** each classified network flow N_l **do**
10. Store result in Local Behavior List (LBL): <Device ID, Behavior, Timestamp>
    Phase 5: LBL Broadcasting
11. Broadcast LBL to:
    Cloud Node (C) for storage and network-wide behavior updates
    Connected Fog Nodes FNs for information sharing
    Phase 6: Global Behavior List (GBL) Formation
12. **for** each FN_k receiving an LBL from another fog node **do**

13. Merge received LBLs to form a Global Behavior List (GBL)
Phase 7: GBL Distribution to IoT Devices
14. Broadcast GBL to IoT devices in communication range via base station
Phase 8: GBL Storage and Update at IoT Devices
15. **for** each IoT device receiving GBL **do**
16. Store GBL in local database
17. Update GBL when a new version is received
18. **END**

4. Results and Discussion

The performance of the models was analyzed using Python that is executed in a system with the following configura tion: AMD Ryzen 7 3750H processor, 16.0 GB of RAM, 1.5 TB of storage, and Windows 10 (64-bit) operating system. In this study, we evaluated the performance of various models using different algorithms across three datasets: UNSW-NB-15, NSL-KDD, and CICIDS-2017. The models considered for performance evaluations consisted of Prototypical Networks, Matching Networks, MAML, Reptile, and Hybrid Model. The performance metrics considered for these per formance evaluations are Accuracy, Precision (P), Recall (R), F1-score, training time, inference time, ROC-AUC curve, energy consumed (watts), cost ($), and processing latency (secs). It is assumed that the above models run on a fog node for attack detection. The fog node operates at a power of 200 watts. Also, the cost of electricity per kWh is considered as $0.12.

- Training time (TT) is the total time taken to train a model with a dataset.

$$TT = ST - ET \tag{22}$$

where, ST is the start time and ET is the end time.

- Inference Time (IT) is the time taken by the trained model to predict the new data.

$$IT = \frac{Total\ time\ \ for\ \ predictions}{Number\ of\ \ Predictions} \tag{23}$$

- ROC-AUC curve is the receiver operating characteristic-area under curve. ROC signifies the graphical representation of TPR against FPR. TPR is the true positive rate and FPR is the false positive rate respectively. AUC shows the integral of the ROC curve.
- Energy consumed is the total energy consumed in watt hours for running the model at the fog node for training and inference.

$$Energy\ Consumed = Energy_{train} + Energy_{infer} \tag{24}$$

$$Energy_{train} = Power_{WATTS} \times time_{train} \tag{25}$$

$$Energy_{infer} = Power_{WATTS} \times time_{infer} \tag{26}$$

- Cost in $ is the total cost required for training and inference.

$$Cost(\$) = Cost_{train} + Cost_{infer} \tag{27}$$

$$Cost_{train} = \frac{Energy_{train}}{1000} \times COST_PER_KWH \tag{28}$$

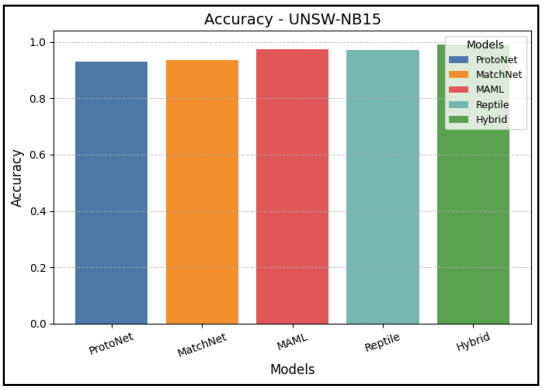
$$Cost_{infer} = \frac{Energy_{infer}}{1000} \times COST_PER_KWH \tag{29}$$

- Processing latency is the total amount of time in seconds (secs) for training and inference.

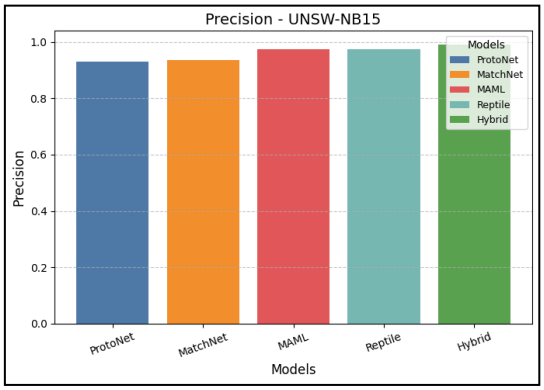
$$Processing\ Latency\ (secs) = Time_{train} + Time_{infer} \tag{30}$$

4.1. Results and Discussion for UNSW-NB15 Dataset

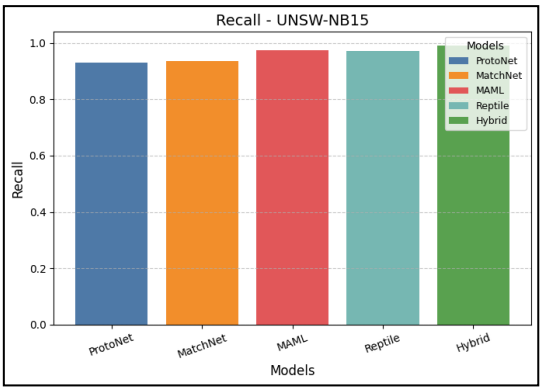
From Fig. 4, it is observed that the proposed MAML-Reptile based hybrid model performed better than Prototypical Networks, Matching Networks with the UNSW-NB-15 dataset in detecting attacks with high accuracy. The performance of the model with respect to the performance metrics considered is listed below. The hybrid model shows accuracy = 0.9903, precision = 0.9904, recall = 0.9903, F1-score = 0.9903, ROC-AUC = 0.9976, training time = 74.01s, and inference time = 0.21s. Similarly, the performance metrics values of Prototypical Network are 0.9300, 0.9306, 0.9300, 0.9301, 0.9785, 14.25s, and 0.13s. Matching Network shows 0.9349, 0.9350, 0.9349, 0.9349, 0.9807, 14.24s, and 0.14s respectively. MAML shows 0.9749, 0.9756, 0.9749, 0.9749, 0.9711, 18.72s, and 0.15s respectively. Reptile shows 0.9730, 0.9737, 0.9730, 0.9729, 0.9710, 18.53s, and 0.26s respectively.



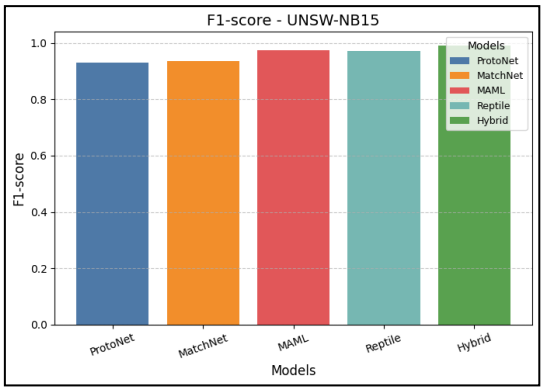
(a). Comparison of accuracy



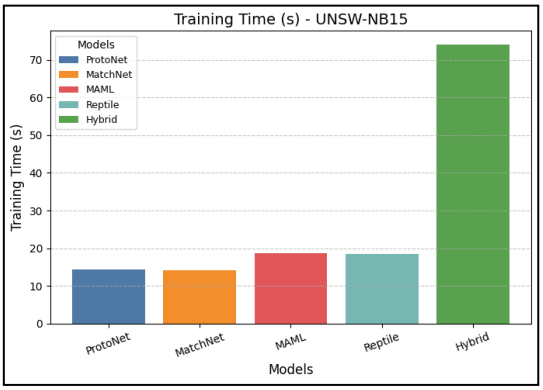
(b). Comparison of precision



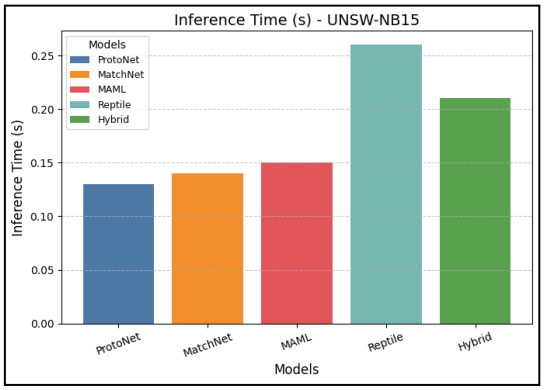
(c). Comparison of recall



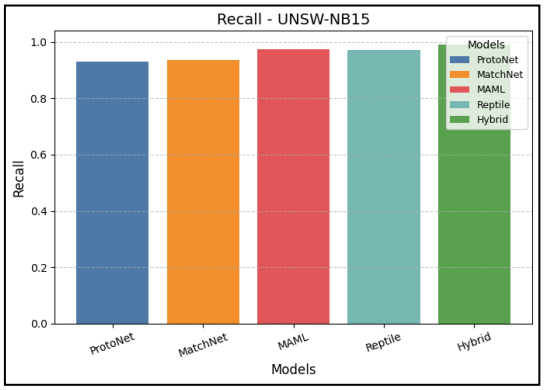
(d). Comparison of F1-Score



(e). Comparison of training time



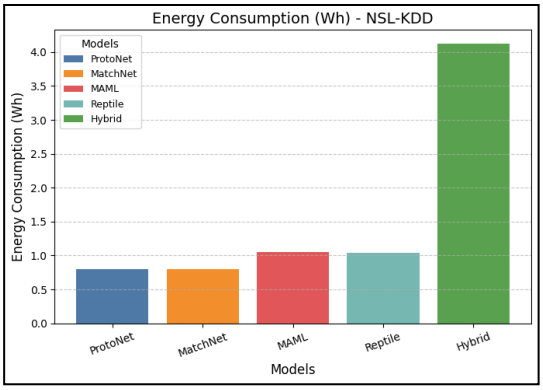
(f). Comparison of inference time



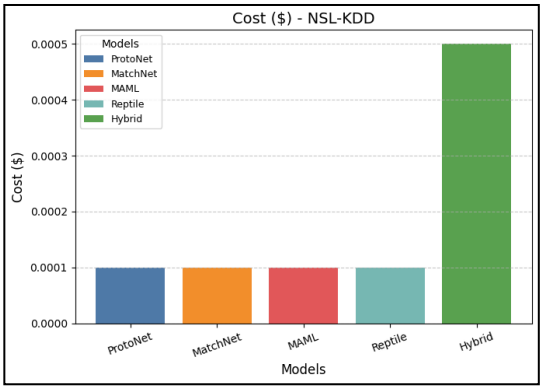
(g). Comparison of ROC-AUC

Fig. 4. Comparison of performance metrics for few-shot learning models on the UNSW-NB-15 dataset.

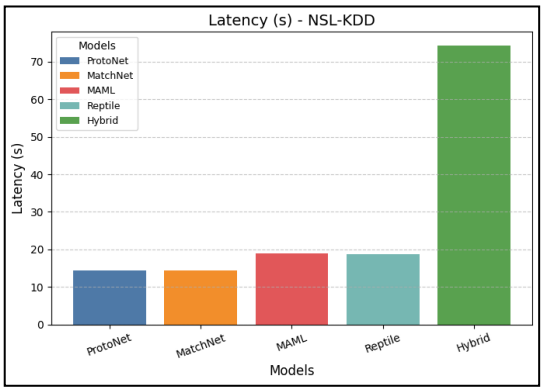
From Fig. 5, it is observed that the proposed hybrid model shows high energy consumption, cost, and processing latency for training and inference. However, it performs better than other few-shot learning models in detecting attacks. Prototypical Network shows energy consumption of 0.80 Watt Hour, cost of $0.0001, and processing latency of 14.39 s. Similarly, Matching Network shows 0.80 Watt Hour, $0.0001, and 14.38s respectively. MAML shows 1.05 Watt Hour, cost $0.0001, and 18.88s respectively. Reptile shows 1.04 Watt Hour, cost of $0.0001 and 18.79s respectively. The hybrid model shows 4.12 Watt Hour, $0.0005, and 74.22s respectively.



(a). Energy consumption in watt hour



(b). Cost in $



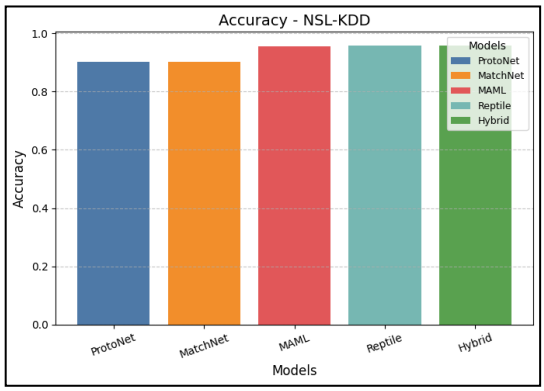
(c). Processing latency in seconds

Fig. 5. Comparison of energy consumption, cost, and processing latency for few-shot learning models on the UNSW-NB15 dataset.

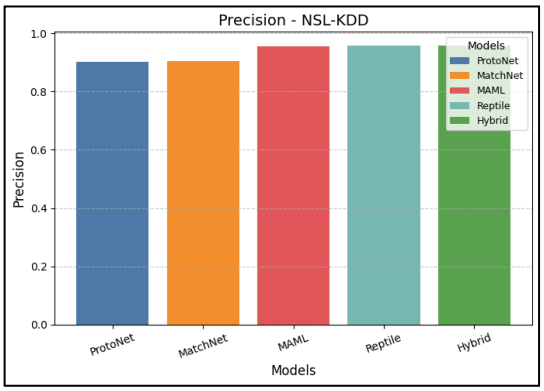
4.2. Results and Discussion for NSL-KDD Dataset

From Fig. 6, it is observed that the proposed MAML-Reptile based hybrid model performed better than Prototypical Networks, Matching Networks with NSL-KDD dataset in detecting attacks with high accuracy. The performance of the model with respect to the performance metrics considered is listed below. However, the hybrid model also performs similarly to the Reptile model. The Reptile model shows accuracy = 0.9572, precision = 0.9573, recall = 0.9572, F1 score = 0.9572, ROC-AUC = 0.9841, Training Time=7.55s, and Inference Time=0.08s. Similarly for these performance metrics hybrid Model shows 0.9568, 0.9573, 0.9568, 0.9567, 0.9843, 32.79s, and 0.09s respectively. Prototypical Network shows 0.9018, 0.9026, 0.9018, 0.9020, 0.9587, 6.98s, and 0.06s respectively.

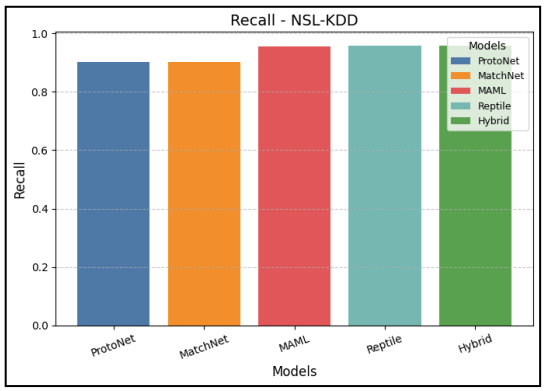
Matching Network shows 0.9031, 0.9038, 0.9031, 0.9034, 0.9576, 5.86s, and 0.06s respectively. MAML shows 0.9543, 0.9543, 0.9543, 0.9543, 0.9822, 8.31s, and 0.07s respectively.



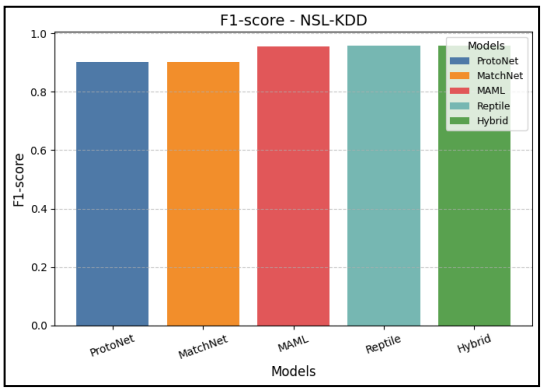
(a). Comparison of accuracy



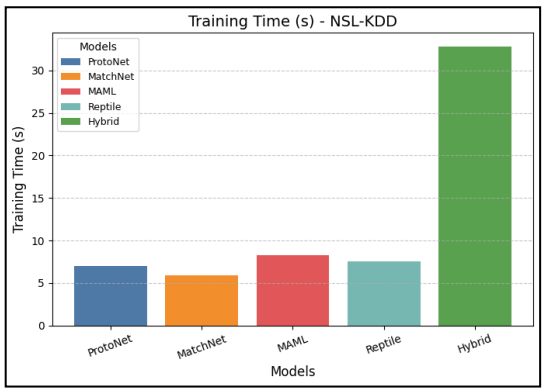
(b). Comparison of precision



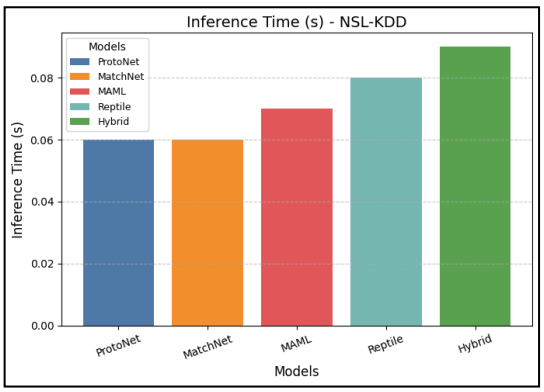
(c). Comparison of recall



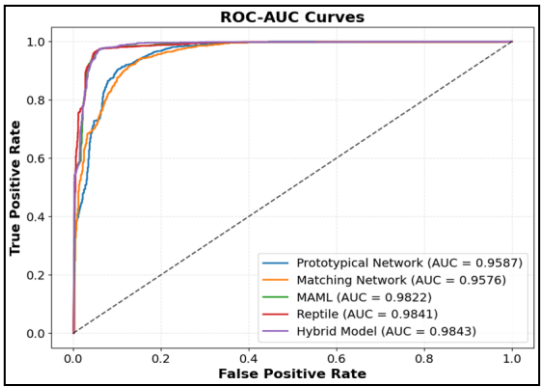
(d). Comparison of F1-score



(e). Comparison of training time



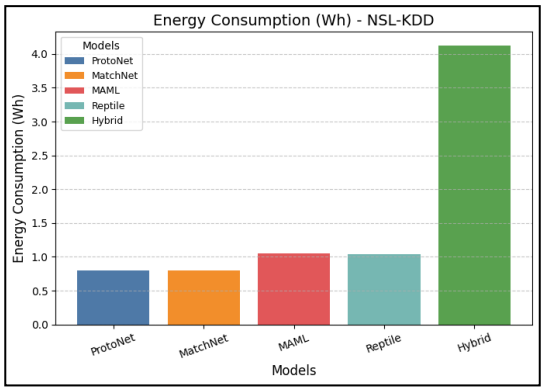
(f). Comparison of inference time



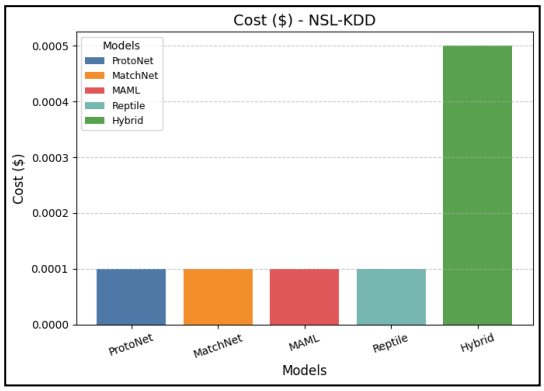
(g). Comparison of ROC-AUC

Fig. 6. Comparison of performance metrics for few-shot learning models on the NSL-KDD dataset.

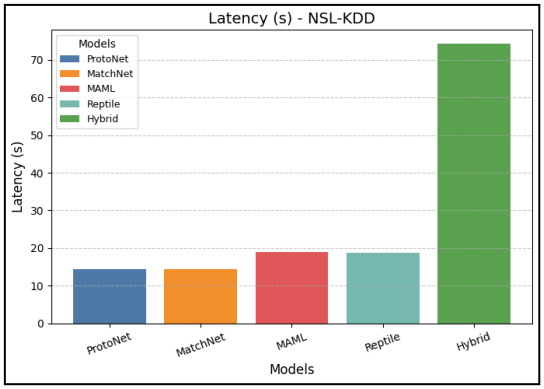
From Fig. 7, it is observed that the hybrid model shows high energy consumption, cost, and processing latency because for training and inference. However, it performs better than other few-shot learning models in detecting attacks. Prototypical Network shows energy consumption of 0.39 Watt Hour, cost of $0.00005, and Processing Latency of 7.04 s. Similarly, Matching Network shows 0.80 Watt Hour, $0.0001, and 14.38s respectively. MAML shows 1.05 Watt Hour, cost $0.0001, and 18.88s respectively. Reptile shows 1.04 Watt Hour, cost of $0.0001 and 18.79s respectively. The hybrid model shows 4.12 Watt Hour, $0.0005, and 74.22s respectively.



(a). Energy consumption in watt hour



(b). Cost in $

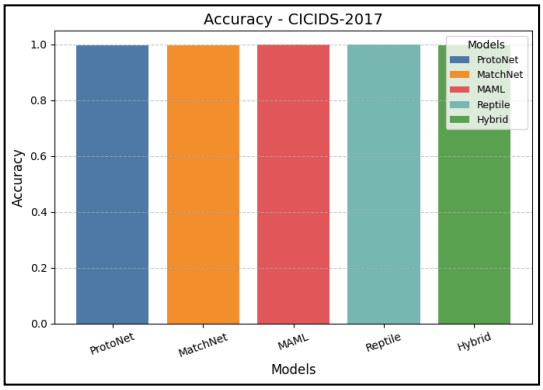


(c). Processing latency in seconds

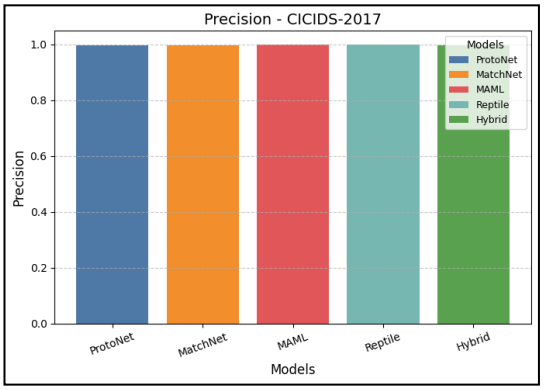
Fig. 7. Comparison of energy consumption, cost, and processing latency for few-shot learning models on the NSL-KDD dataset.

4.3. Results and Discussion for CICIDS-2017 Dataset

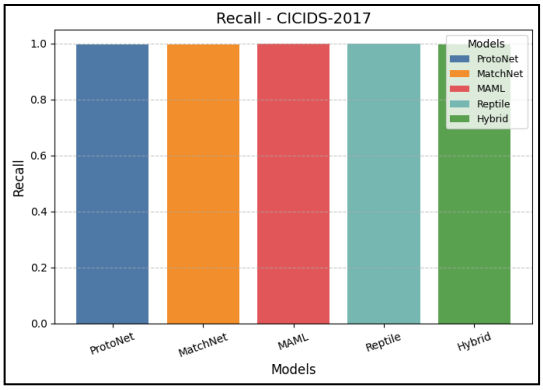
From Fig. 8, it is observed that the proposed MAML-Reptile based hybrid model performed similar to Prototypical Networks, Matching Networks with the CICIDS-2017 dataset in detecting attacks with high accuracy. However, the hybrid model and all other models also performed similarly to the MAML model. MAML shows accuracy = 0.9989, precision = 0.9989, recall = 0.9989, F1-score = 0.9989, ROC-AUC = 0.9994, training time = 92.92s and inference time = 0.70s. Similarly, the Hybrid Model shows 0.9979, 0.9979, 0.9979, 0.9979, 0.9997, 474.27s, and 2.53s respectively. Prototypical Network shows 0.9982, 0.9982, 0.9982, 0.9982, 0.9998, 72.09s, and 0.76s respectively. Matching Network shows 0.9983, 0.9983, 0.9983, 0.9983, 0.9998, 68.93s, and 0.65s respectively. Reptile shows 0.9988, 0.9988, 0.9988, 0.9988, 0.9994, 95.42s, and 0.70s.



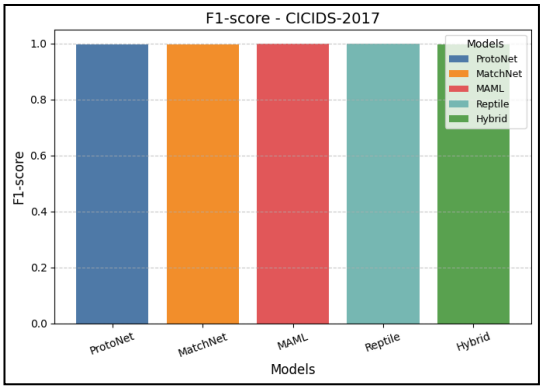
(a). Comparison of accuracy



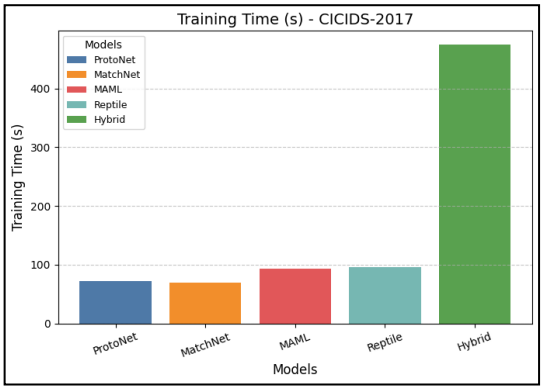
(b). Comparison of precision



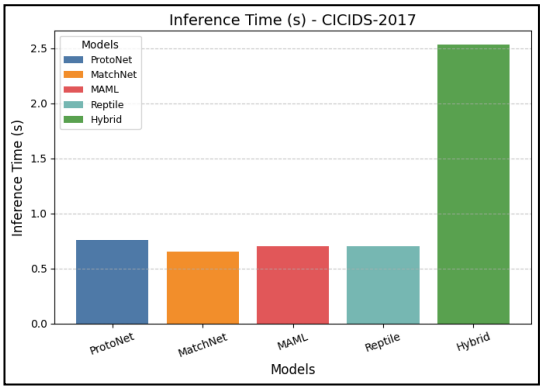
(c). Comparison of recall



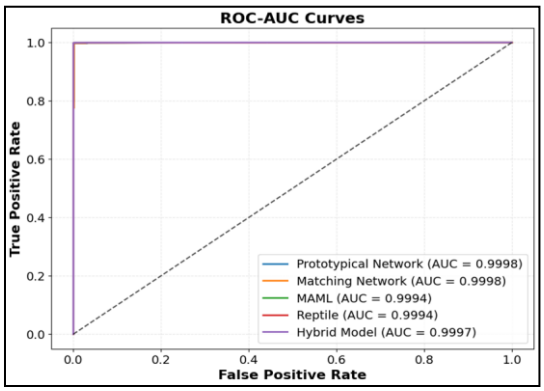
(d). Comparison of F1-Score



(e). Comparison of training time



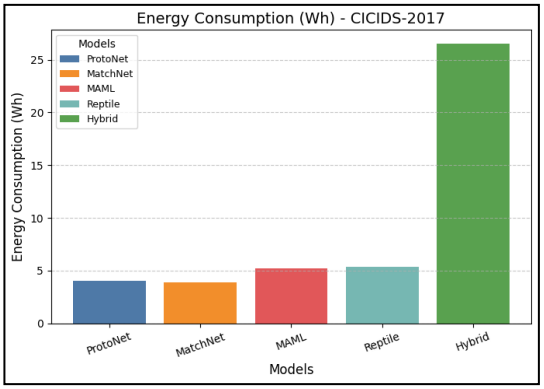
(f). Comparison of inference time



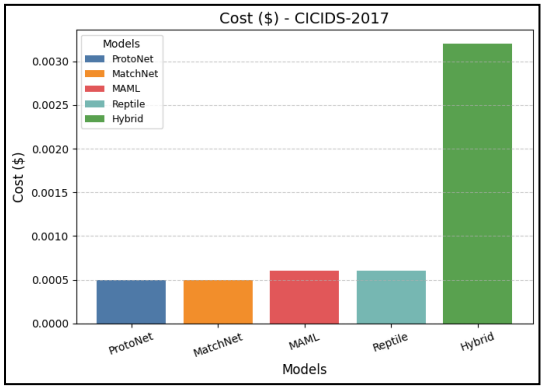
(g). Comparison of ROC-AUC

Fig. 8. Comparison of performance metrics for few-shot learning models on the CICIDS-2017 dataset.

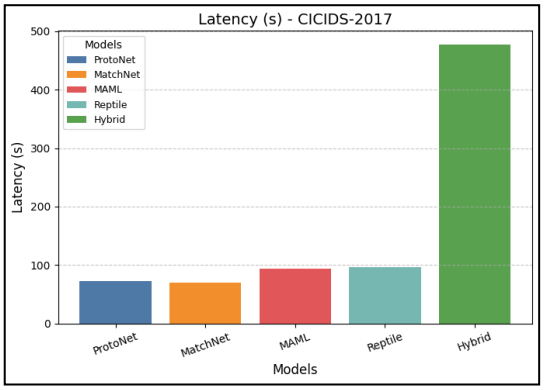
From Fig. 9, it is observed that the proposed hybrid model shows high energy consumption, cost, and processing latency for training and inference. However, it performs better than other few-shot learning models in detecting attacks. Prototypical Network shows energy consumption of 4.05 Watt Hour, cost of $0.0005, and processing latency of 72.85 s. Similarly Matching Network shows 3.87 Watt Hour, $0.0005, and 69.58s respectively. MAML shows 5.20 Watt Hour, cost $0.0006, and 93.62s respectively. Reptile shows 5.34 Watt Hour, cost of $0.0006 and 96.12s. Hybrid model shows 26.49 Watt Hour, $0.0032, and 476.80s. Table 1 and Table 2 show the whole comparison of the models.



(a). Energy consumption in watt hour



(b). Cost in $



(c). Processing latency in seconds

Fig. 9. Comparison of energy consumption, cost, and processing latency for few-shot learning models on the CICIDS-2017 dataset.

4.4. Discussion

The results in Table 1 show that the hybrid model does not always provide the same level of improvement across datasets. For UNSW-NB15 and NSL-KDD, it shows higher accuracy and ROC-AUC than most of the single models, which indicates that combining meta-learning approaches helps when the traffic is more diverse or imbalanced. In contrast, for CICIDS 2017 all single models already perform almost perfectly. In this case, the hybrid model gives only a very small difference while its training time is much higher. This means that for practical use, the hybrid is most useful when the dataset is complex and harder to learn. But, when the dataset is already clear and separable, single models are sufficient and more efficient. The results also highlight the limitations of the hybrid model in terms of efficiency. While it shows the best accuracy on UNSW-NB15 and near-best accuracy on NSL-KDD, its energy consumption and latency are much higher than the single models. For example, in UNSW-NB15 the hybrid uses about 4.12 Wh with 74s latency. For MAML or Reptile, it is much lower. The gap is even larger in CICIDS-2017, where the hybrid model consumes 26.49 Wh with latency 476.80s. This means that the hybrid is not always practical for IoT and fog environments where devices have limited energy and low delay requirements. In such cases, single models like Prototypical or Matching Networks are more efficient models with high accuracy. The hybrid is more suitable for powerful edge servers or cloud deployments, where higher resource use can be tolerated with strong detection performance. Also, this can be a good future direction to design a lightweight model. However, the proposed model shows high accuracy in detecting maleficent behaviors in Fog-IoT networks.

Table 1. Performance comparison of few-shot learning models on different datasets.

Dataset	Model	Accuracy	Precision	Recall	F1-score	ROC-AUC	Training Time (s)	Inference Time (s)
UNSW-NB15	Prototypical Network	0.93	0.9306	0.93	0.9301	0.9785	14.25	0.13
	Matching Network	0.9349	0.935	0.9349	0.9349	0.9807	14.24	0.14
	MAML	0.9749	0.9756	0.9749	0.9749	0.9711	18.72	0.15
	Reptile	0.973	0.9737	0.973	0.9729	0.971	18.53	0.26
	Hybrid Model	0.9903	0.9904	0.9903	0.9903	0.9976	74.01	0.21
NSL-KDD	Prototypical Network	0.9018	0.9026	0.9018	0.902	0.9587	6.98	0.06
	Matching Network	0.9031	0.9038	0.9031	0.9034	0.9576	5.86	0.06

	MAML	0.9543	0.9543	0.9543	0.9543	0.9822	8.31	0.07
	Reptile	0.9572	0.9573	0.9572	0.9572	0.9841	7.55	0.08
	Hybrid Model	0.9568	0.9573	0.9568	0.9567	0.9843	32.79	0.09
CICIDS-2017	Prototypical Network	0.9982	0.9982	0.9982	0.9982	0.9998	72.09	0.76
	Matching Network	0.9983	0.9983	0.9983	0.9983	0.9998	68.93	0.65
	MAML	0.9989	0.9989	0.9989	0.9994	0.9994	92.92	0.7
	Reptile	0.9988	0.9988	0.9988	0.9994	0.9994	95.42	0.7
	Hybrid Model	0.9979	0.9979	0.9979	0.9997	0.9997	474.27	2.53

Table 2. Comparison of energy consumption, cost, and processing latency for few-shot learning models.

Dataset	Model	Energy Consumption (Watt Hour)	Cost ($)	Processing Latency (s)
UNSW-NB15	Prototypical Network	0.8	0.0001	14.39
	Matching Network	0.8	0.0001	14.38
	MAML	1.05	0.0001	18.88
	Reptile	1.04	0.0001	18.79
	Hybrid Model	4.12	0.0005	74.22
NSL-KDD	Prototypical Network	0.8	0.0001	14.39
	Matching Network	0.8	0.0001	14.38
	MAML	1.05	0.0001	18.88
	Reptile	1.04	0.0001	18.79
	Hybrid Model	4.12	0.0005	74.22
CICIDS-2017	Prototypical Network	4.05	0.0005	72.85
	Matching Network	3.87	0.0005	69.58
	MAML	5.2	0.0006	93.62
	Reptile	5.34	0.0006	96.12
	Hybrid Model	26.49	0.0032	476.8

5. Conclusions

This work presents the design of a hybrid MAML-Reptile based few-shot learning approach to secure fog-based IoT networks against maleficent behaviors. The few-shot models taken for performance comparison are Prototypical Networks, Matching Networks, MAML, and Reptile. The fog node is deployed with the best model to detect attacks and broadcasts the LBL and GBL to share more attack information to IoT devices. Here, we consider standard datasets such as UNSW-NB-15, NSL-KDD, and CICIDS-2017 to perform evaluations. From the results, it is observed that the proposed MAML-Reptile hybrid model performed better in terms of accuracy, recall, F1-score, precision, and ROC-AUC than other standard models in detecting maleficent behaviors. However, due to the hybridization of the model, the training time, cost, energy consumption, and processing latency increase. In the future, the evaluations of the proposed hybrid model can be extended to edge devices to design a lightweight hybrid few-shot learning model with adaptive tuning of hyperparameters (τ and ζ) to dynamically balance accuracy and efficiency. In addition, the real-world deployment can be studied to measure scalability.

All the Declarations and Statements

Author Contributions Statement

Surya Pavan Kumar Gudla – Conceptualization, Literature Survey, Methodology, Implementation, Result Analysis, Manuscript Writing and Editing.
Sourav Kumar Bhoi– Problem Finding, Literature Survey, Methodology, Result validation, Supervision, Manuscript Writing and Editing.
Kshira Sagar Sahoo – Model Training, Validation, and Performance Evaluation, Manuscript Writing and Editing.
GNV Rajareddy – Result Analysis, Manuscript Editing.

All authors have read and agreed to the published version of the manuscript.

Conflict of Interest Statement
The authors declare no conflicts of interest.

Funding Declaration
There is no funding information.

Data Availability Statement
The data is available on request.

Ethical Declarations
Not Applicable.

Acknowledgments
We sincerely thank BPUT Rourkela and PMEC Berhampur for providing the infrastructure to conduct this research work.

Declaration of Generative AI in Scholarly Writing
Not Applicable.

Abbreviations
The following abbreviations are used in this manuscript:

MAML - Model Agnostic Meta Learning
LBL - Local Behavior List
GBL - Global Behavior List
IoT - Internet of Things
C - Cloud Node
FN - Fog Node
TT - Training Time
IT - Inference Time

References

[1] Shanhe Yi, Cheng Li, and Qun Li. A survey of fog computing: concepts, applications and issues. In Proceedings of the 2015 workshop on mobile big data, pages 37–42, 2015.
[2] Hany F Atlam, Robert J Walters, and Gary B Wills. Fog computing and the internet of things: A review. big data and cognitive computing, 2(2):10, 2018.
[3] Ivan Stojmenovic, Sheng Wen, Xinyi Huang, and Hao Luan. An overview of fog computing and its security issues. Concurrency and Computation: Practice and Experience, 28(10):2991–3005, 2016.
[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pages 13–16, 2012.
[5] Amir Vahid Dastjerdi, Harshit Gupta, Rodrigo N Calheiros, Soumya K Ghosh, and Rajkumar Buyya. Fog comput ing: Principles, architectures, and applications. In Internet of things, pages 61–75. Elsevier, 2016.
[6] G Shruthi, Monica R Mundada, S Supreeth, and Bryan Gardiner. Deep learning-based resource prediction and mutated leader algorithm enabled load balancing in fog computing. International Journal of computer networks and information security, 15(4):84–95, 2023.
[7] Tamara Al-Shurbaji, Mohammed Anbar, Selvakumar Manickam, Iznan H Hasbullah, Nadia ALfriehate, Basim Ah mad Alabsi, Ahmad Reda Alzighaibi, and Hasan Hashim. Deep learning-based intrusion detection system for detecting iot botnet attacks: A review. IEEE Access, 2025.
[8] SamaaYTarabay, AbeerTwakol, AhmedSSamrah, andYasserIbrahim. Asecure and efficient cryptography system based on chaotic maps for securing data image in fog computing. International Journal of Computer Network and Information Security, 15(1):64–80, 2023.
[9] Dipti Prava Sahu, Biswajit Tripathy, and Leena Samantaray. Optimized intrusion detection system in fog computing environment using automatic termination-based whale optimization with elm. International Journal of Computer Network and Information Security, 16(2):79–91, 2024.
[10] Yifan Wang, Tetsutaro Uehara, and Ryoichi Sasaki. Fog computing: Issues and challenges in security and forensics. In 2015 IEEE 39th annual computer software and applications conference, volume 3, pages 53–59. IEEE, 2015.
[11] Saad Khan, Simon Parkinson, and Yongrui Qin. Fog computing security: a review of current applications and security solutions. Journal of Cloud Computing, 6:1–22, 2017.

[12] Shruti Jadon and Aryan Jadon. An overview of deep learning architectures in few-shot learning domain. arXiv preprint arXiv:2008.06365, 2020.
[13] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1199–1208, 2018.
[14] Md Moin Uddin Chowdhury, Frederick Hammond, Glenn Konowicz, Chunsheng Xin, Hongyi Wu, and Jiang Li. A few-shot deep learning approach for improved intrusion detection. In 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), pages 456–462. IEEE, 2017.
[15] Merna Gamal, Hala M Abbas, Nour Moustafa, Elena Sitnikova, and Rowayda A Sadek. Few-shot learning for discovering anomalous behaviors in edge networks. Computers, Materials and Continua, 69(2):1823–1837, 2021.
[16] MDIslam, Aminu Yusuf, Muhammad Dikko Gambo, and Abdulaziz Y Barnawi. A novel few-shot ml approach for intrusion detection in iot. Arabian Journal for Science and Engineering, pages 1–15, 2024.
[17] Ahmed Samy, Haining Yu, and Hongli Zhang. Fog-based attack detection framework for internet of things using deep learning. Ieee Access, 8:74571–74585, 2020.
[18] Ahmed Demirpolat, Alper Kaan Sarica, and Pelin Angin. Prot ´edge: a few-shot ensemble learning approach to software-defined networking-assisted edge security. Transactions on Emerging Telecommunications Technologies, 32(6):e4138, 2021.
[19] Thin Tharaphe Thein, Yoshiaki Shiraishi, and Masakatu Morii. Few-shot learning-based malicious iot traffic detec tion with prototypical graph neural networks. IEICE TRANSACTIONS on Information and Systems, 106(9):1480 1489, 2023.
[20] Jingcheng Yang, Hongwei Li, Shuo Shao, Futai Zou, and Yue Wu. Fs-ids: A framework for intrusion detection based on few-shot learning. Computers & Security, 122:102899, 2022.
[21] Mostafa Hosseini and Wei Shi. Intrusion detection in iot network using few-shot class incremental learning. In Future of Information and Communication Conference, pages 617–636. Springer, 2024.
[22] Abebe Diro and Naveen Chilamkurti. Leveraging lstm networks for attack detection in fog-to-things communica tions. IEEE Communications Magazine, 56(9):124–130, 2018.
[23] Dina Ayesha S and Siddique AB. Fs3: few-shot and self-supervised framework for efficient intrusion detection in internet of things networks. In Proceedings of the 39th Annual Computer Security Applications Conference, pages 138–149, 2023.
[24] Shizhao Tian, Zhen Wang, Kaiyu Xie, Fansong Chen, and Hongsong Zhu. Fog-enabled intrusion detection method integrating bi-lstm and multi-head self-attention for iot. In 2024 27th International Conference on Computer Sup ported Cooperative Work in Design (CSCWD), pages 2004–2009. IEEE, 2024.
[25] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. Advances in neural information processing systems, 30, 2017.
[26] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. Advances in neural information processing systems, 29, 2016.
[27] Rinu Boney and Alexander Ilin. Semi-supervised few-shot learning with maml. 2018.
[28] Uday Kulkarni, Meena SM, Raghavendra Hallyal, Prasanna Sulibhavi, Shankru Guggari, and Akshay R Shanbhag. Optimisation of deep neural network model using reptile meta learning approach. Cognitive Computation and Systems, 2023.
[29] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw nb15 network data set). In 2015 military communications and information systems conference (MilCIS), pages 1–6. IEEE, 2015.
[30] L Dhanabal and SP Shantharajah. A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. International journal of advanced research in computer and communication engineering, 4(6):446–452, 2015.
[31] Ranjit Panigrahi and Samarjeet Borah. A detailed analysis of cicids2017 dataset for designing intrusion detection systems. International Journal of Engineering & Technology, 7(3.24):479–482, 2018.

Authors' Profiles



Surya Pavan Kumar Gudla is currently pursuing his Ph.D. from Department of Computer Science and En gineering, Biju Patnaik University of Technology (a state govt. university), Rourkela, India. He received his MCA and MTech in CSE Engg from Jawaharlal Nehru Technological University Kakinada (JNTUK), India. He is currently working as Asst.Prof. in Department of Computer Science and Engineering Aditya Institute of Technology and Management (AITAM), Tekkali, Andhra Pradesh. His interested areas are Computer Networks, Mobile Computing and DataMining. He has published 15 research papers in reputed international journals and conferences.



Sourav Kumar Bhoi (Senior Member, IEEE and Member, ACM) Sourav Kumar Bhoi received the Ph.D. degree and M.Tech degree from the Department of Computer Science and Engineering, National Institute of Technology (NIT), Rourkela, India, in 2017 and 2013, respectively. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Parala Maharaja Engineering College (a Government Engineering College), Berhampur, India. He has nearly ten years of teaching and research experience. He also completed his Post Doc in CSIS programme during 2022-23 from SU, MAN. He also completed a three-month CSIR Summer Research Training Program in online mode from NEIST, Jorhat, Assam (Government of India), in August 2020. He acted as Project investigators for research projects sponsored by MHRD and AICTE. He is an India Book of Record Holder for his research work, in 2022. He has more than 140 research publications in reputed international journals, conferences, books, book chapters, technical articles, patents, and theses. His

research interests include machine learning, the Internet of Things, edge and fog computing, ad hoc and sensor networks, and information security. He is a member of many professional bodies, such as a member of IAENG, a Life Member of CSI, an Associate Member of IEI, and a fellow of SIESRP. He acted as a member of TPC and the session chair of many international conferences. He received the prestigious IET Premium Award from IET Networks journal, in 2016. He also received many other awards and honors, such as the University Foundation Day Faculty Research Award in CSE and the Sadananda Memorial Award from the Institution of Engineers (India), in 2021 and 2020, respectively. He was a reviewer for many international journals and conferences. He delivers several invited talks in reputed organizations.



Kshira Sagar Sahoo (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from the National Institute of Technology (NIT), Rourkela, India, in 2019 and master's degree in information and communication technology from the Indian Institute of Technology (IIT), Kharagpur, India, in 2014. He is currently working as an Assistant Professor in the Department of Computer Science and Engineering at SRM University, Andhra Pradesh, India. He was a post doctoral Kempe fellow with the Autonomous Distributed Systems Laboratory, Department of Computing Science, Ume˚a University, Ume˚a Sweden from Oct 2022 to Sept 2024. His research interests include SDN, edge computing, IoT, Industrial IoT, 5G, urban anomaly detection, and machine learning. He has published more than 80 research papers in the leading international journals and conferences. He is a Senior Member of the IEEE Computer Society and an Associate Member of the Institute of Engineers India.



Gnv Rajareddy received the Ph.D. degree from National Institute of Technology, Silchar, India in 2023. He is currently a Faculty in Department of Chemical and Biological Engineering, University of Saskatchewan, Saskatoon, Canada. He has eight years of full-time teaching experience. He has published five research articles in various reputed international journals. His research interests include image processing and deep learning. He is a member of IAENG, UACEE, and IFERP.

How to cite this paper:

Surya Pavan Kumar Gudla, Sourav Kumar Bhoi, Kshira Sagar Sahoo, GNV Rajareddy, "A Hybrid MAML-Reptile Based Few-shot Learning Approach for Securing Fog-IoT Networks against Maleficent Behaviors ", International Journal of Computer Network and Information Security(IJCNIS), Vol.18, No.3, pp. 44-62, 2026. DOI:10.5815/ijcnis.2026.03.03