Cognitive IoT and Edge Computing for Intrusion Detection with Federated TinyML

Mingyan Li
Oak Ridge National Laboratory
lim3@ornl.gov

Paul Laiu
Oak Ridge National Laboratory
liaump@ornl.gov

Jeff A. Nichols
Oak Ridge National Laboratory
nicholsja2@ornl.gov

Mike Huettel
Oak Ridge National Laboratory
huettelmr@ornl.gov

Isaac Sikkema
Oak Ridge National Laboratory
sikkemaic@ornl.gov

Mahim Mathur
Oak Ridge National Laboratory
mathurm@ornl.gov

Sam Hollifield
Oak Ridge National Laboratory
hollifieldsc@ornl.gov

Max Hankins
Oak Ridge National Laboratory
hankinswm@ornl.gov

***Abstract*—Internet of Things (IoT) and Edge Computing (EC) are rapidly becoming an integral part of the modern society. By 2030, there is estimated to be over 40 billion active and connected IoT devices [1]. This rapid progress also comes with a significant implication on cybersecurity. Back-end infrastructure and systems have a much broader attack than they did previously due to vulnerable IoT/EC devices being connected to wireless networks. This expanding attack surface is a growing concern because IoT/EC are increasingly being used in critical systems such as power grids, health care, and smart homes. To effectively address a problem of this scale, cognitive cyber methods—which can autonomously detect and react to cyber attacks as they develop—are needed. To address this, we bring Artificial Intelligence (AI) and Machine Learning (ML) to IoT/EC devices, using tinyML to monitor voluminous IoT data against cyber threats, and using Federated Learning (FL) to share local detection knowledge across the system while preserving privacy. We propose a novel three-layer architecture: (1) an IoT layer for tinyML-based inference, (2) an edge layer for ML model training, and (3) a cloud layer for FL operations. Using the publicly available 11-class N-BaIoT dataset [2], we demonstrate that this architecture mitigates resource constraints at the IoT layer while improving detection accuracy over standard two-layer designs. An outlier-resistant scaler, feature reduction, and quantization enable the tinyML model to maintain detection accuracy with a reduced model size. Additionally, federated learning that only utilizes the intersection (across heterogenous devices) of the reduced feature set achieves superior detection accuracy compared to locally trained models.**

***Index Terms*—federated learning, tinyML, IoT intrusion detection, cognitive cyber, IoT security architecture**

This manuscript has been co-authored by UT-Battelle, LLC, under contract DEAC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

I. Introduction

The ubiquity of IoT/EC devices underscores their critical role in our computing infrastructure, fundamentally driving unprecedented levels of connectivity and efficiency in our daily lives. However, out of the 40 billion IoT devices expected to be connected by 2030, over half may be vulnerable to cyber-attacks [3]. IoT/EC vulnerabilities are being exploited in sophisticated multi-step campaigns with devastating effects that spread across the broader system of systems (e.g., data centers or smart grid ecosystems). For example, the 2013 Target data breach [4] that led to large-scale cyber fraud and compromised more than 41 million customer accounts. The multi-step campaign leveraged a virus compromise of Point of Sales IoT/EC devices, illustrating how IoT/EC vulnerabilities allow adversaries to bypass security of a large-scale enterprise even when traditional intrusion routes are well-defended. Another example is Mirai malware and its variants that launched distributed denial-of-service (DDOS) attacks from IoT almost crippling the Internet in 2016, endangering both critical systems and human lives [5].

Detection of IoT/EC-level attacks remains a challenge since it needs to be performed at the local device to enable low-latency decision making and privacy of local network traffic. For example, the N-BaIoT [2] dataset contains both normal and attack network traffic from nine commercial IoT devices during Mirai and Bashlite family botnet attacks. Principal Component Analysis on N-BaIoT shows observable network traffic differs from device to device [6], due to IoT device-to-device communication patterns and intrinsic characteristics differences (e.g., manufacturers, types, functionalities). This

visibility constraint implies detection must take place locally, yet today's IoT lacks intelligence to recognize threats, especially zero-day exploits. In addition to the challenge of on-device intelligent detection, yet another challenge is to accurately process a myriad of diverse, multi-mode data across heterogeneous IoT [7]. This leads us to examine AI/ML approaches to help address the issues.

The Internet of Things Cybersecurity Improvement Act of 2020 (Public Law No: 116-207, H.R. 1668) mandates National Institute of Standards and Technology (NIST) and the Office of Management and Budget (OMB) to enhance IoT cybersecurity. As IoT/EC adoption expands across critical infrastructure, including smart grids and distributed energy resources (DER), ensuring robust security measures becomes increasingly vital. Even IoT-enabled smart homes, as extensions of larger networked systems, introduce new attack vectors. Our goal is to provide cognitive cyber detection and adaptive response capabilities to address these emerging threats..

II. Related Work

A. TinyML

TinyML, an approach that enables ML on ultra-low power devices, has emerged as a highly active research area [8]. The goal of applying tinyML on IoT devices is to meet the memory and computational limitation of IoT/MCU devices, while maintaining models' accuracy [9].

While tinyML has witnessed applications in diverse fields such as healthcare, agriculture, transportation, and surveillance [10], its potential in cyber threat detection and network traffic anomaly detection has been relatively underexplored [11], [12]. In [11], a Naïve Bayes based tinyML model is employed to predict threats from IoT log data, in conjunction with MITRE's published cybersecurity vulnerabilities. Another study [12] compares the energy consumption of various ML models in cloud, edge, and tinyML inferences at MCUs for IoT intrusion detection. Other works also explored compressed ML models to realize on-device intrusion detection without using the term "tinyML" [6], [13]. However, none of these works treats the device and data heterogeneity issue DRIFT faces. As noted in [8], it is challenging to adopt a specific tinyML model in a heterogeneous IoT environment, and this requires more research.

B. Federated Learning

Federated learning (FL) [14]–[16] is a machine learning paradigm that builds a global predictive model on a central server from data distributed on multiple local devices. On the central server, FL iteratively aggregates local models trained with the computation resource on the local devices to obtain the global model, which is expected to be more generalizable and robust than each of the local models after the aggregation. FL differs from traditional distributed optimization in that FL does not require sharing the local data on each device with either the server or other local devices and that the privacy of each local dataset should be protected. In the context of cybersecurity, keeping the data local implies the reduction of attack surface and communication cost. Given the growing number of IoT and edge devices, the communication cost could easily be the computational bottleneck for training machine learning models from data distributed on these devices, which makes FL well-suited for realizing practical machine learning assisted cybersecurity enhancement tools. This feature also enables FL to perform machine learning tasks on private, sensitive data. In fact, many recent works on FL have been focusing on the protection of data privacy [17]–[19], including healthcare data [20], [21], audio recording data [22], [23] and household smart meter data [24], [25]. Applying FL to detect cybersecurity attacks on distributed systems has recently gained attention [26], [27]. Existing work has shown that FL is able to provide a global model that outperforms the local models. Many of the existing work use variants of a classical FL method, Federated averaging (FedAvg) [28], which computes the global model parameters as a weighted average of the local model parameters and thus requires that the local ML models share the same architecture. Therefore, for tinyML models that share both model architecture and prediction tasks, standard FL techniques are well-positioned that we expect them to give a boost in model accuracy over each individually trained model.

C. Federated Learning with tinyML for IoT

Existing works targeting resource-constrained devices, such as IoT or edge devices, often focus on either tinyML or FL independently, as noted in [29]. While recent research has explored the intersection of tinyML and FL, none have directly addressed intrusion detection in IoT or edge devices.

For example, Ren et al. leverage FL and meta-learning to adapt an image recognition neural network (NN) for audio keyword recognition in a tinyML setting [30]. This work highlights the adaptability of models across domains, but does not consider security applications in IoT. Similarly, Ficco et al. focus on local classification and regression tasks for IoT devices using tinyML on an ECG heartbeat dataset, with FL facilitating privacy-preserving cloud-based training [31]. Their work uniquely combines FL with transfer learning, to enable the reuse of FL-enhanced models for device-specific tasks. Kopparapu et al. focus on an open-source FL implementation designed for resource-constrained IoT devices, such as microcontroller units (MCUs) and small CPU-based devices and demonstrates the feasibility of FL for distributed learning in IoT environments [32]. A gap exists applying Federated tinyML to IoT intrusion detection, which is the focus of this paper.

III. DRIFT Architecture

By utilizing tinyML and the collective learning framework FL, our research premise is that DRIFT will enhance IoT devices' cognitive intelligence for smart detection and adaptive defense, thus improving overall IoT network resilience and security against cyber-attacks. Figure 1 illustrates DRIFT operates at three layers of infrastructure abstraction: IoT/MCU, edge, and cloud/enterprise. We differentiate the more powerful edge layer (e.g., home routers, gateways) from the less capable

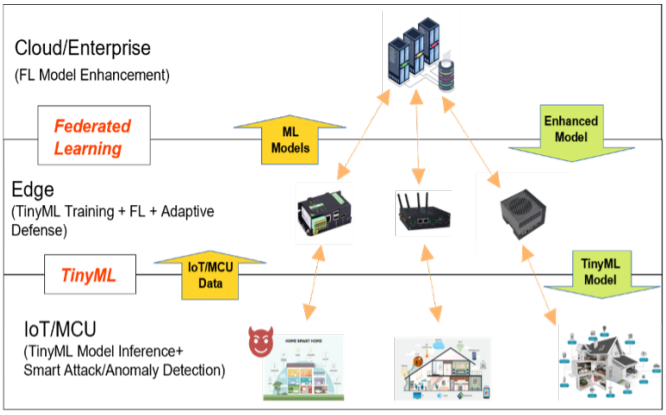

Fig. 1. DRIFT conceptual architecture at three layers: IoT/MCU, edge, and cloud/Enterprise

IoT/MCU devices (e.g., smart appliances) as a majority of today's tinyML approaches perform computation intensive ML training on edge devices and less demanding trained-models inference on IoT/MCU devices. We further envision and design DRIFT to execute adaptive knowledge-based defense countermeasures at the edge level

- At the lowest IoT/MCU layer where cyber-attacks occur, multiple devices run tinyML models monitoring network traffic and neighbor activities to detect anomalies and attacks. The data are sent to the edge layer for training.
- In the middle edge layer, tinyML models are trained and optimized. The models are passed downward to IoT/MCU for inference execution detecting attacks, and they are also passed upward to FL for global model enhancement and re-distribution. Edge also performs adaptive knowledge-based responses against detected anomalies and/or attacks.
- In the cloud/enterprise layer, FL aggregates multitude of tinyML models, generates, and re-distributes enhanced global models to achieve detection knowledge propagation and improve system-side detection performance.

Advantages of the DRIFT three-layer architecture: The DRIFT architecture comprises an FL server at the cloud layer and FL clients at the edge layer. Each edge node (i.e., an FL client) aggregates data from IoT/MCU devices within its local environment and trains a model on the collected data. By offloading local training from IoT/MCU devices to resource-rich edge nodes, the DRIFT architecture not only alleviates the resource constraints of IoT devices but also reduces communication bandwidth with the FL server, when compared to the standard two-layer FL architecture where IoT devices act as FL clients [29], [31]. Additionally, this approach has the potential to improve the detection accuracy due to the aggregated data at the edges, as demonstrated by the experimental results in Section V-B2.

A. TinyML

In the context of Internet of Things (IoT) and Microcontroller Unit (MCU) layers, there are inherent limitations in computational resources, including processing power, memory, and storage. IoT devices, especially those based on MCUs, often lack the processing capacity needed for complex machine learning models, making them unsuitable for direct model training. Consequently, model training is typically performed at the edge layer, where devices with more computational resources are available. This edge layer can include gateways, servers, or even cloud platforms that aggregate and process data from IoT devices, allowing for more computationally intensive tasks like model training.

However, even after training at the edge, the resulting models must be deployed back onto IoT devices for real-time inference, and this introduces another challenge: model size. Although the model might perform well on the edge, it must be small and efficient enough to fit the memory and processing limitations of the IoT device during inference.

To address this, we explore feature selection methods in this paper for model reduction. Compared to model optimization techniques such as quantization, pruning, and knowledge distillation, proper feature selection not only can reduce the complexity and size of the model, but can also save computational energy in constructing and preprocessing (such as normalization) of these features. Feature selection techniques, such as those based on statistical metrics or tree-based methods, help identify and retain only the most informative features, leading to a smaller model that can be efficiently deployed on resource-constrained devices. This reduction not only optimizes the inference speed but also reduces the memory footprint, making it feasible for deployment on IoT/MCU layers with minimal computational overhead. In our work, we explore various feature selection strategies to enable the creation of smaller, more efficient models that can perform well on resource-limited IoT devices without sacrificing accuracy.

B. Federated learning

In the DRIFT architecture, FL is performed between the edge layer with the edge devices as distributed clients and the cloud/enterprise layer with a global server which aggregates the tinyML models trained on individual edge devices. FL is an essential component in the DRIFT architecture because it can propagate knowledge about malicious network traffic between edge devices without sharing potentially private or sensitive traffic data. This advantage becomes particularly important, as demonstrated in the experiments in Section V-B2, when different edge devices observe different types of attacks, which is a common scenario in a distributed environment.

The heterogeneity of device data poses unique challenges in FL algorithms. For example, the well-known client drift effect shows that the deviation of the global model learned in FL from the one learned in a centralized setting increases as the data become more heterogeneous among the distributed devices. In this work, we investigate the impacts on model accuracy from scalers and feature selections in the data preprocessing stage when the data are heterogeneous. These factors are often overlooked in FL tests on well-established benchmarks, such as MNIST, CIFAR10, and ImageNet, for

which proper global preprocessing steps are easily obtainable from prior knowledge. However, in the network traffic data considered in this work, a unified preprocessor is often not available apriori. With results from the study of data preprocessing for network traffic data, we demonstrate the effectiveness of FL on the DRIFT architecture in the numerical experiments reported in Section V-B2.

IV. TinyML Feature selection

A. Network traffic features

Temporal-statistical features are essential in network traffic analysis for intrusion detection, as they help identify abnormal patterns in traffic behavior that may signal malicious activities. For example, attackers often exhibit distinct traffic patterns, such as sudden spikes in packet frequency or irregular communication bursts, which differ from normal behavior.

The Kitsune feature extraction [33] is an on-line, scalable approach to calculate temporal statistics from network traffic flows. It employs a damped window to maintain incremental statistics over time, where older statistics are gradually phased out and recent changes in network behavior manifest. This method is memory-efficient as it allows outdated statistics to be discarded once their dampening weight reaches zero, reducing the memory footprint for high-speed data processing. We employed Kisune feature extraction on raw pcap files to generated 115 features, i.e., 23 features in each of the five different dampening windows. The features include multiple statistical metrics such as mean, jitter, covariance of the arriving packet' source MAC and IP address, source IP address, source and destination IP address pair, and source and destination TCP/UDP socket pair.

B. Feature selection algorithms

In this study, we use four different feature selection methods to identify the most relevant features for intrusion detection in IoT networks as follows.

1) Mutual Information: Mutual information (1) measures the amount of information shared between two variables, quantifying how much knowing one reduces variable uncertainty about the other. It is especially useful for detecting non-linear relationships between features and the target. The higher mutual information between a feature and the target, the more important a feature to predict the target.

$$MI(X,Y) = H(X) + H(Y) - H(X,Y) \tag{1}$$

where $H(X), H(Y)$ are the marginal entropy of X(features) and Y(target) and $H(X,Y)$ si the joint entropy.

2) Analysis of Variance (ANOVA) F-statistics: The ANOVA F-statistic tests whether the means of multiple groups are significantly different. The method compares the variance between the different class groups to the variance within each group.

$$F = \frac{\text{Between-group variance}}{\text{Within-group variance}}$$

The between-group variance captures the variance of the class means relative to the overall mean, while the within-group variance measures the compactness within each class. A higher F-statistic with a feature indicates a a larger distance between the classes with a smaller variance within a class. Therefore, the higher the F statistics, the better thetter the discriminative power the feature has between classes.

3) Tree-based feature importance: Tree-based models like Random Forest or Gradient Boosting Machines assign feature importance by measuring how much each feature reduces the model's impurity or error during the splitting process. Impurity refers to a measure of data heterogeneity, the lower impurity indicates more homogeneity, i.e., more data points belong to the same set. For example, Gini impurity at a splitting node is defined as $1-\sum_{i=1}^{C} {p_i}^2$ where p_i is the proportion of samples of class i in the node and C is the number of classes. Each tree in the ensemble splits the data using features that reduce impurity, and the importance of a feature is calculated based on the average reduction in impurity across all trees.

4) Shapley Values: Shapley values, derived from cooperative game theory, provide a fair method to allocate the contribution of each characteristic to the prediction of the model of the model [34]. The values consider all possible subsets of features and compute the marginal contribution of each feature to the prediction.

$$\phi_i(v) = \frac{1}{n} \sum_{S \subseteq N \setminus \{i\}} \binom{n-1}{|S|}^{-1} (v(S \cup \{i\}) - v(S))$$

where $\phi_i(v)$ is the Shapley value for feature i, $v(s)$ is the model's prediction for a subset of feature S, N denotes the set of all features.

The Shapley value computes the average contribution of each feature over all possible combinations, accounting for interactions between features. This makes it a comprehensive and interpretable method for the importance of features.

V. Numerical/Experimental results

A. Problem setting

1) Dataset: In this paper, we demonstrate the performance of the DRIFT architecture on network attack classification tasks from network traffics. We consider the N-BaIoT dataset [2], which contains network traffic data from nine different IoT devices, including doorbells, baby monitors, and security cameras with ten different types of network attacks labeled together with the regular benign traffic. The ten types of attacks are from two IoT malware families: Mirai and BASHLITE (Gafgyt), each of which is further categorized into five variants based on their attack methods. For Mirai, there are Scan (botnet robes during the reconnaissance), ACK (ACK flood), Syn (Syn flood), UDP (UDP flood) and UDPplain (UDP flood without payload) attacks data, while BASHLITE attack data include Scan, Combo (hybrid TCP/ACK/PSH flood), Junk (floods with random service), UDP, and TCP flood data. The N-BaIoT dataset comprises 115 statistical features constructed from raw network traffic using Kitsune feature extraction method [33] as described in Section IV-A. The N-BaIoT dataset is widely used to train ML models to detect IoT

botnet attacks and serves as a benchmark to evaluate intrusion detection systems targeting such threats [6], [35].

2) Model and training configuration: To ensure that the learned classifier can be trained and deployed on the edge devices, we use a small multilayer perceptron (MLP) with only one hidden layer of width 100 as the classifier. This results in 12,711 parameters (∼60 KB) when all 115 features are used. In the feature selection experiments in Section V-B1, the Multi-Layer Perceptron (MLP) classifier is implemented and trained using the `scikit-learn` package [36] in Python, whereas in the FL experiments in Section V-B2, the MLP is implemented in `TensorFlow` [37], the federated environment is simulated using the `Flower` framework [38], and the model quantization is performed under the `TensorFlow Lite` framework. In all tests, the MLPs are trained with the Adam optimizer with a constant learning rate 0.001. The categorical cross-entropy loss function is used in the training, where as the MLP performance metric is the categorical classification accuracy.

B. Numerical experiments

1) Preprocessing and feature selection: This section investigates the features in the N-BaIoT dataset to improve the accuracy and reduce the size of MLP classification models, which builds the foundation of the FL study in the next section.

In all tests considered in this work, the data are preprocessed by applying a quantile transformer (`QuentileTransformer()` in `scikit-learn`) with a standard normal (Gaussian) distribution to each feature. This choice is motivated from the fact that the outliers in the N-BaIoT dataset can severely skew the commonly used standard scaler (`StandardScaler()` in `scikit-learn`), resulting in inaccurate classification for MLP models. Figure 2 illustrates the confusion matrices from MLP models trained on data preprocessed with the standard scaler (left) and the quantile transformer (right).

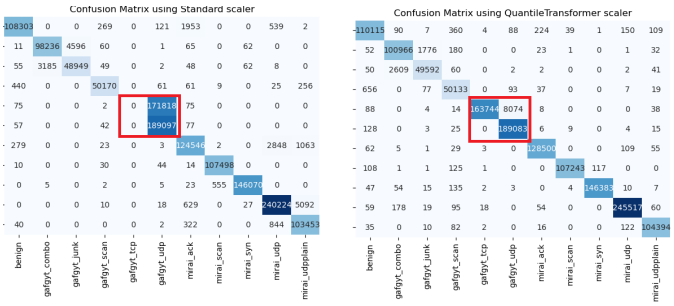

Fig. 2. Confusion matrices of MLP models with a standard scaler (left) and a quantile transformer (right). By using an outlier-resistant quantile transformer, the MLP is able to differentiates between gafgyt_tcp and gafgyt_udp attacks when a standard scaler fails.

As shown in Figure 3, among the four evaluated feature selection methods, the Random Forest (RF) importance-based method performed the best, achieving 95% accuracy with only five features. In contrast, ANOVA yielded the poorest results. Although Shapley value-based feature selection (FS) achieved an accuracy comparable to mutual information-based FS, it requires a preliminary model (similar to tree-based importance FS) and involves costly Shapley value computations. Therefore, for our FL experiments, we selected the tree-based importance method (highest accuracy) and mutual information-based FS (second-best accuracy with lower computational overhead).

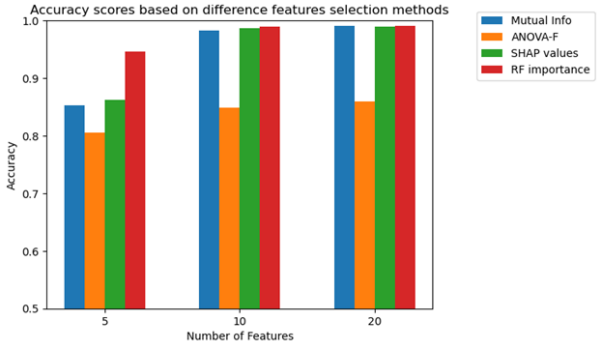

Fig. 3. Comparison of accuracy using four feature selection methods.

In the FL tests in Section V-B2, we consider both the mutual information-based FS and the RF importance-based FS methods to reduce the number of features used in FL. The details of FS in the FL setting are discussed in the next section.

2) Federated learning: In this section, we demonstrate the advantage of the DRIFT architecture by performing FL on the N-BaIoT dataset for network attack classification. We consider the scenario that the nine IoT devices in the N-BaIoT dataset are partitioned into three groups, each connects to an EC device. FL is then performed among the three EC devices (EC1, EC2, and EC3) and a server to learn a global classifier for network attack classification. The connection between the EC devices and the IoT devices in the N-BaIoT dataset is given in Table I, where we numbered the IoT devices considered in the N-BaIoT dataset as IoT1, ..., IoT9, to simplify the presentation.

EC1	IoT1	Danmini_Doorbell
	IoT2	Philips_B120N10_Baby_Monitor
	IoT3	SimpleHome_XCS7_1002_WHT_Security_Camera
	IoT4	SimpleHome_XCS7_1003_WHT_Security_Camera
EC2	IoT5	Ennio_Doorbell
	IoT6	Samsung_SNH_1011_N_Webcam
EC3	IoT7	Ecobee_Thermostat
	IoT8	Provision_PT_737E_Security_Camera
	IoT9	Provision_PT_838_Security_Camera

TABLE I

APPLICATION OF DRIFT ARCHITECTURE TO THE IOT DEVICES CONSIDERED IN THE N-BAIOT DATASET — THREE EDGE DEVICES ARE CONNECTED TO FOUR, TWO, AND THREE IOT DEVICES, RESPECTIVELY.

This choice of EC-IoT connection is to promote the heterogeneity of local data on each EC. Specifically, both IoT5 and IoT6 connected to EC2 only see the benign and Gafgyt types of traffics while missing Mirai types of attacks as shown in Table II, where the total number of local data collected from each IoT device and the ratio of traffic classes are reported.

traffic class	IoT1	IoT2	IoT3	IoT4	IoT5	IoT6	IoT7	IoT8	IoT9
benign	4.9%	16.0%	5.4%	2.3%	11.0%	13.9%	1.6%	7.5%	11.8%
gafgyt_scan	2.9%	2.5%	3.2%	3.4%	7.9%	7.4%	3.3%	3.5%	3.4%
gafgyt_tcp	9.0%	8.4%	10.3%	11.5%	28.6%	26.1%	11.4%	12.6%	10.7%
gafgyt_udp	10.4%	9.6%	12.0%	12.1%	29.2%	29.5%	12.5%	12.6%	12.5%
gafgyt_junk	2.9%	2.6%	3.3%	3.2%	8.4%	7.5%	3.6%	3.7%	3.5%
gafgyt_combo	5.9%	5.3%	6.3%	7.0%	14.9%	15.6%	6.3%	7.4%	6.9%
mirai_syn	12.0%	10.8%	14.6%	14.4%	0.0%	0.0%	14.0%	7.9%	7.4%
mirai_scan	10.6%	9.4%	5.3%	5.1%	0.0%	0.0%	5.2%	11.7%	11.6%
mirai_udpplain	8.1%	7.4%	9.1%	9.9%	0.0%	0.0%	10.5%	6.8%	6.4%
mirai_ack	10.0%	8.3%	12.9%	12.6%	0.0%	0.0%	13.6%	7.3%	6.9%
mirai_udp	23.3%	19.8%	17.6%	18.5%	0.0%	0.0%	18.1%	18.9%	19.0%
# of data	1.02M	1.10M	0.86M	0.85M	0.36M	0.38M	0.84M	0.83M	0.84M

TABLE II

LOCAL IOT TRAFFIC DATA AMOUNT AND TRAFFIC-CLASS-DISTRIBUTION IN THE N-BAIOT DATASET — MOST DEVICES OBSERVE ALL 11 CLASSES OF NETWORK TRAFFIC, WITH THE EXCEPTION OF IOT5 AND IOT6, WHICH DO NOT OBSERVE ANY MIRAI-TYPE TRAFFICS.

Feature selection	Accuracy	Local training			FL
		EC1	EC2	EC3	
all features	local val.	99.54%	99.85%	99.66%	99.87%
	global test	96.51%	45.33%	96.79%	99.75%
mutual information	local val.	99.62%	99.38%	99.55%	98.45%
	global test	92.45%	44.03%	96.73%	97.35%
RF importance	local val.	99.54%	98.92%	99.44%	97.42%
	global test	91.82%	45.71%	96.49%	96.06%

TABLE III

TINYML MODEL ACCURACY COMPARISON IN DRIFT ARCHITECTURE — CLASSIFICATION ACCURACY OF LOCALLY TRAINED MLP MODELS AND FL MLP MODELS ON BOTH THE LOCAL VALIDATION AND THE GLOBAL TEST DATASETS ARE REPORTED.

We compare the accuracy of MLP classifiers trained on each EC devices with only the local data to the ones of MLP classifiers learned by applying FL to the three EC devices. In these tests, the local data on each device are split into disjoint training, validation, and test datasets via stratified uniformly random sampling based on the traffic class label. The sizes of the validation and test datasets are 20% and 10% of the corresponding local datasets, respectively. As for the training datasets, instead of using the remaining 70% of the data, we only use 0.1% of the local data in the training. This choice is justified by the nearly perfect classification results on the validation and test datasets reported in this section.[1] A global test dataset is then constructed by jointing the testing datasets from all nine IoT devices. The global test dataset includes network traffic data for all 11 classes and is of size 10% of the entire N-BaIoT dataset. We use the classification accuracy on this global test set as the primary metric of the model performance.

In these tests, all feature are preprocessed with a quantile transformer with standard normal (Gaussian) distribution as described in Section V-B1 to mitigate the impact of outliers. To avoid leakage of global information, we apply the quantile transformer fit for the training dataset on EC1 to process the data on each EC device. We also consider three FS approaches – (i) selecting all features, (ii) mutual information-based FS, and (iii) RF importance-based FS – in the tests. For (ii) and (iii), FS is performed on each of the IoT devices and select 40 features. To ensure the features on each FL client are consistent, we take the intersection of the 40 features selected on each of the IoT devices as the final selected feature set. This process results in 26 and 24 (out of 115) features for the mutual information-based and RF importance-based FS methods, respectively.

The classification accuracy of the MLP models trained locally or via FL are given in Table III, where both the accuracies on the associated local validation dataset and on the global test dataset are reported. The local MLP models (columns EC1, EC2, and EC3 in Table III) are trained with 50 epochs under configurations described in Section V-A2. In the FL setting (column FL in Table III), the global MLP models are trained via FL with 50 communication/aggregation rounds. Each communication round includes the local update stage, in which the three EC devices take one local epoch to optimize the local MLP model based on local training data, and the aggregation stage, in which the EC devices communicate the updated local model parameters to the server for aggregation. In this work, we use the FedAvg algorithm proposed in [28] as the FL algorithm, where the aggregation is a simple weighted averaging of the local model weights.

The results in Table III show that, even with nearly perfect local validation accuracies, the local MLP models trained on EC2 has significantly lower global test accuracy, due to the missing Mirai attack classes on the local EC2 dataset. The FL results demonstrate that FL can aggregate local information and leads to superior global test accuracy than models trained locally. This conclusion holds for all three feature selection approaches considered in this work.

We further investigate the scenario that when each of the IoT devices is directly connected to the server, i.e., when there is no EC layer. In this case, FL is performed across the nine IoT devices directly without passing through EC devices. The MLP classifiers are trained both locally and via FL in this standard two-layer architecture under the same training configuration as the tests on the DRIFT architecture reported in Table III. The resulting model accuracy are reported in Table IV. Observations from Table IV include (i) IoT5 and IoT6 still have much lower global test accuracy than local validation accuracy, due to the lack of Mirai related traffic data, (ii) all IoT devices show global test accuracy than local validation accuracy indicating an increase of data heterogeneity in this scenario, (iii) the FL MLP model accuracy is noticeably higher than the ones from local training, and (iv) the resulting FL models are not as accurate as their counterparts trained under the DRIFT architecture in Table III, especially when feature selection is incorporated.

Finally, we perform quantization of the learned FL models to further compress the model size of deployments on MCUs. The quantization process converts MLP parameters from 32 bit floats to 8 bit integers via converters provided in the `TensorFlow Lite` package. The resulting model size and performance are reported in Table V, which shows that model quantization results in around 4x compression as expected with minimal impact to the global test accuracy. This result

[1]The fact that all 11 classes can be well-classified by using only 0.1% of the data indicates potential data redundancy in the N-BaIoT dataset, which warrants further investigation but is beyond the scope of the current work.

Feature selection	Accuracy	Local training									FL
		IoT1	IoT2	IoT3	IoT4	IoT5	IoT6	IoT7	IoT8	IoT9	
all features	local val.	99.16%	100.0%	98.48%	98.48%	99.07%	96.91%	98.32%	98.15%	99.49%	97.70%
	global test	77.43%	86.03%	82.16%	73.97%	42.21%	36.14%	81.35%	78.27%	83.54%	97.52%
mutual information	local val.	98.48%	98.65%	96.46%	96.80%	98.46%	97.53%	97.64%	97.81%	97.14%	91.19%
	global test	77.65%	79.23%	77.83%	81.79%	46.10%	42.02%	80.27%	82.70%	80.41%	89.37%
RF importance	local val.	98.32%	99.33%	97.64%	97.31%	98.77%	98.77%	98.15%	98.82%	98.82%	89.84%
	global test	78.30%	78.00%	80.79%	82.60%	45.59%	42.59%	81.06%	77.56%	78.96%	86.81%

TABLE IV

TINYML MODEL ACCURACY COMPARISON IN STANDARD TWO-LAYER ARCHITECTURE.

indicates that model quantization is an effective approach to compress the model size when the memory on MCU is limited.

model	all features		mutual information		RF importance	
	acc.	size	acc.	size	acc.	size
original	99.75%	62KB	97.35%	27KB	96.06%	26KB
quantized	99.38%	15KB	97.32%	6KB	95.82%	6KB

TABLE V

MODEL QUANTIZATION RESULTS IN 4X COMPRESSION OF MODEL SIZE WITH MINIMAL IMPACTS ON THE GLOBAL TEST ACCURACY.

VI. CONCLUSION AND DISCUSSION

To integrate AI/ML into IoT/EC for effective cybersecurity intrusion detection, we presented and validated a three-layer (tinyML + FL) architecture that achieves higher botnet detection accuracy than local models alone, using the 11-class N-BaIoT dataset. Our approach demonstrated that a quantized, feature-reduced tinyML model on IoT devices preserves detection accuracy, while FL leveraging the intersection of reduced feature sets maintains detection accuracy. The IoT network attack classification performance was enhanced utilizing an outlier resistant scaler. While our paper validated this three-layer intelligent IoT/EC approach over cybersecurity intrusion detection, we also foresee the applicability of such multi-layer (tinyML+FL) framework effective over additional application domains such as smart grid, smart factory monitoring, and systematic sensing. Future research will assess real-time deployment factors such as latency and energy use, along with the model's robustness to varying conditions and its scalability in complex environments.

REFERENCES

[1] N. Kumar, "How Many IoT Devices Are There (2025-2030 Data)," Dec. 2024.

[2] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, Jul. 2018.

[3] "More Than Half of IoT Devices Vulnerable to Severe Attacks," https://threatpost.com/half-iot-devices-vulnerable-severe-attacks/153609/, Mar. 2020.

[4] "What We Learned from Target's Data Breach 2013 | CardConnect," https://www.cardconnect.com/launchpointe/payment-trends/target-data-breach/.

[5] "The 5 Worst Examples of IoT Hacking and Vulnerabilities in Recorded History," https://www.iotforall.com/5-worst-iot-hacking-vulnerabilities.

[6] B. Sudharsan, D. Sundaram, P. Patel, J. G. Breslin, and M. I. Ali, "Edge2Guard: Botnet Attacks Detecting Offline Models for Resource-Constrained IoT Devices," in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and Other Affiliated Events (PerCom Workshops)*, Mar. 2021, pp. 680–685.

[7] C. R. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov, D. Patterson, D. Pau, J.-s. Seo, J. Sieracki, U. Thakker, M. Verhelst, and P. Yadav, "Benchmarking TinyML Systems: Challenges and Direction," Jan. 2021.

[8] R. Kallimani, K. Pai, P. Raghuwanshi, S. Iyer, and O. L. A. López, "TinyML: Tools, applications, challenges, and future research directions," *Multimedia Tools and Applications*, vol. 83, no. 10, pp. 29 015–29 045, Mar. 2024.

[9] D. L. Dutta and S. Bharali, "TinyML Meets IoT: A Comprehensive Survey," *Internet of Things*, vol. 16, p. 100461, Dec. 2021.

[10] M. Shafique, T. Theocharides, V. J. Reddy, and B. Murmann, "TinyML: Current Progress, Research Challenges, and Future Roadmap," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, Dec. 2021, pp. 1303–1306.

[11] A. Dutta and S. Kant, "Implementation of Cyber Threat Intelligence Platform on Internet of Things (IoT) using TinyML Approach for Deceiving Cyber Invasion," in *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, Oct. 2021, pp. 1–6.

[12] N. Tekin, A. Acar, A. Aris, A. S. Uluagac, and V. C. Gungor, "Energy consumption of on-device machine learning models for IoT intrusion detection," *Internet of Things*, vol. 21, p. 100670, Apr. 2023.

[13] R. Bekkouche, M. Omar, R. Langar, and B. Hamdaoui, "Ultra-Lightweight and Secure Intrusion Detection System for Massive-IoT Networks," in *ICC 2022 - IEEE International Conference on Communications*, May 2022, pp. 5719–5724.

[14] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[15] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[16] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *International conference on machine learning*. PMLR, 2014, pp. 1000–1008.

[17] Z. Li, J. Zhang, L. Liu, and J. Liu, "Auditing privacy defenses in federated learning via generative gradient leakage," *arXiv preprint arXiv:2203.15696*, 2022.

[18] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 937–16 947, 2020.

[19] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[20] O. Choudhury, Y. Park, T. Salonidis, A. Gkoulalas-Divanis, I. Sylla *et al.*, "Predicting adverse drug reactions on distributed health data using federated learning," in *AMIA Annual symposium proceedings*, vol. 2019. American Medical Informatics Association, 2019, p. 313.

[21] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *International journal of medical informatics*, vol. 112, pp. 59–67, 2018.

[22] F. Granqvist, M. Seigel, R. van Dalen, Á. Cahill, S. Shum, and M. Paulik, "Improving on-device speaker verification using federated learning with privacy," *arXiv preprint arXiv:2008.02651*, 2020.

[23] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," in *ICASSP 2019-2019 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 6341–6345.

[24] Y. Wang, I. L. Bennani, X. Liu, M. Sun, and Y. Zhou, "Electricity consumer characteristics identification: A federated learning approach," *IEEE Transactions on Smart Grid*, vol. 12, no. 4, pp. 3637–3647, 2021.

[25] A. Taïk and S. Cherkaoui, "Electrical load forecasting using edge computing and federated learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[26] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, "Federated Learning for Intrusion Detection System: Concepts, Challenges and Future Directions," Jun. 2021, comment: Submitted to JNCA, Elsevier.

[27] X. Sáez-de-Cámara, J. L. Flores, C. Arellano, A. Urbieta, and U. Zurutuza, "Clustered Federated Learning Architecture for Network Anomaly Detection in Large Scale Heterogeneous IoT Networks," Mar. 2023.

[28] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[29] P. Qi, D. Chiaro, and F. Piccialli, "Small models, big impact: A review on the power of lightweight Federated Learning," *Future Generation Computer Systems*, vol. 162, p. 107484, Jan. 2025.

[30] H. Ren, D. Anicic, and T. A. Runkler, "TinyReptile: TinyML with Federated Meta-Learning," in *2023 International Joint Conference on Neural Networks (IJCNN)*, Jun. 2023, pp. 1–9.

[31] M. Ficco, A. Guerriero, E. Milite, F. Palmieri, R. Pietrantuono, and S. Russo, "Federated learning for IoT devices: Enhancing TinyML with on-board training," *Information Fusion*, vol. 104, p. 102189, Apr. 2024.

[32] K. Kopparapu, E. Lin, J. G. Breslin, and B. Sudharsan, "TinyFedTL: Federated Transfer Learning on Ubiquitous Tiny IoT Devices," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and Other Affiliated Events (PerCom Workshops)*, Mar. 2022, pp. 79–81.

[33] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," in *Proceedings 2018 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, 2018.

[34] H. Wang, Q. Liang, J. T. Hancock, and T. M. Khoshgoftaar, "Feature selection strategies: A comparative analysis of SHAP-value and importance-based methods," *Journal of Big Data*, vol. 11, no. 1, p. 44, Mar. 2024.

[35] M. S. Ahmad and S. M. Shah, "A lightweight mini-batch federated learning approach for attack detection in IoT," *Internet of Things*, vol. 25, p. 101088, Apr. 2024.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[37] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[38] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.