IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 17, 2020, accepted February 6, 2021, date of publication February 11, 2021, date of current version February 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3058528

LocKedge: Low-Complexity Cyberattack Detection in IoT Edge Computing

TRUONG THU HUONG[1], (Member, IEEE), TA PHUONG BAC[1], DAO M. LONG[1], BUI D. THANG[1], NGUYEN T. BINH[1], TRAN D. LUONG[1], AND TRAN KIM PHUC[2]

[1]School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi 11615, Vietnam
[2]GEMTEX Laboratory, Ecole Nationale Supérieure des Arts et Industries Textiles, 59056 Roubaix, France

Corresponding author: Truong Thu Huong (huong.truongthu@hust.edu.vn)

This work was supported by the Hanoi University of Science and Technology (HUST) under Project T2020-SAHEP-010.

ABSTRACT Internet of Things (IoT) and its applications are becoming commonplace with more devices, but always at risk of network security. It is therefore crucial for an IoT network design to identify attackers accurately, quickly and promptly. Many solutions have been proposed, mainly concerning secure IoT architectures and classification algorithms, but none of them have paid enough attention to reducing the complexity. Our proposal in this paper is an edge-cloud architecture that fulfills the detection task right at the edge layer, near the source of the attacks for quick response, versatility, as well as reducing the Cloud's workload. We also propose a multi-attack detection mechanism called LocKedge (Low-Complexity Cyberattack Detection in IoT Edge Computing), which has low complexity for deployment at the edge zone while still maintaining high accuracy. LocKedge is implemented in two manners: centralized manner and federated learning manner in order to verify the performance of the architecture from different perspectives. The performance of our proposed mechanism is compared with that of other machine learning and deep learning methods using the most updated BoT-IoT data set. The results show that LocKedge outperforms other algorithms such as NN, CNN, RNN, KNN, SVM, KNN, RF and Decision Tree in terms of accuracy and NN in terms of complexity.

INDEX TERMS IoT, security, multi-class detection, feature processing, federated learning, deep learning.

I. INTRODUCTION

The Internet of Things (IoT) is a system of interconnected devices that can transfer data automatically through a network to provide services. In recent years, IoT has been emerged with a lot of potential applications such as healthcare, agriculture, logistics, and urban management. Along with this come a lot of challenges, as the distributed and heterogeneous nature of IoT allows various attacks such as DoS, DDoS, spyware, phishing, etc. Thus, a reliable IoT system must meet many security requirements such as access control and authentication at the edge layer and attack detection at the network layer [1].

However, as IoT systems get larger with more and more connecting devices, bringing in much more traffic - with an estimate of 43 billion IoT devices by 2023 [2] and of 500 billion IoT devices by 2030. It gets even harder to deal with those functions while keeping the system's response fast or in real-time. Therefore, there is a need for reducing the complexity of any attack detection process [3].

In a flexible and scalable IoT platform, the central cloud computing provides large storage and enough computing capacity to process data collected from IoT devices. However, offloading computationally intensive tasks to a cloud center may result in a delay, due to the time needed to transmit, process, and receive a large amount of data. To overcome this limitation, edge computing was born to quickly perform the necessary computational task in the network edge.

Typically, the attack detection technique is done at the network layer, while the edge layer, which is expected to have lower computing power, deals with authentication, limited access, threat hunting and data encryption. With more powerful edge devices nowadays, migrating the detection function to the edge can obviously reduce communication time as well as the Cloud's workload, which might be essential in applications that involve a large amount of simultaneous of

The associate editor coordinating the review of this manuscript and approving it for publication was Giovanni Pau.

users such as traffic monitoring in a smart city. Furthermore, as the source of attacks such as DDoS and Mirai botnet is mostly from compromised end devices, putting attack detection at the edge – which is closer to end devices than the Cloud – will result in faster reaction time. Also, from the privacy perspective, IoT devices may not want to send their data far into the Cloud, but prefer local processing. Therefore attack detection might be more efficient with data that can be only accessed on the edge. However, detection techniques normally employed at the Cloud cannot simply be moved to the edge due to the computing capacity. Mechanisms must be more light-weighted at the edge while still maintaining high performances.

To tackle this problem, we propose an edge-cloud based security scheme – LocKedge (Low-Complexity Cyberattack Detection in IoT Edge Computing) that retains the advantages of the centralized cloud and the edge. In LocKedge, we enable 2 modes: Centralized learning mode and Federated learning mode. In the centralized learning mode, the Cloud receives information of the entire network so that it can do the training phase and update the training model to edge nodes. While the edge carries on the threat detection task, so that the processing intelligence is performed near to the data sources. In the federated learning mode, edge nodes carry on the detection procedure as well the training task; but in this mode, the edge sends simplified information such as weights for the Cloud to update the training model globally. In fact, in terms of attack detection, edge computing can reduce the communication time between the Edge and the Cloud, thereby increasing the system response during attacks, being cost-effective to process and analyze data without network communications. Edge computing also reduces the workload of the central cloud.

LocKedge is a detection framework for multiple types of attacks. LocKedge utilizes the traditional neural network (NN), but it is more light-weighted thanks to deploying some techniques to reduce the dimension of the data before the detection phase, the number of layers and the number of neurons in the hidden layer to minimize the solution's complexity while keeping the detection performance high. Therefore, LocKedge with its high accuracy and low complexity can be suitable for deployment at edge devices with limited computational capacity. On another hand, we also investigate LocKedge in two manners: centralized learning and federated learning manner to train the system for attack detection. The investigation helps us to have more insight into the performance of the IoT architecture and computing capacity.

The rest of this paper is structured as follows. Section II presents the review of the literature. Section III describes our edge-cloud based LocKedge detection framework. The attack detection mechanism in this framework is discussed in section IV. Section V evaluates the mathematical complexity of our algorithm as well as its performance against under attack. We also evaluate the impact of attack volumes in the computing performance and resource consumption of the Edge in a case study. Finally, Section VI is for the conclusion and future work.

II. RELATED WORK

Research in security for the IoT networks is increasingly expanding in terms of both security architectures and mechanisms. We can find a group of authors who follow the SDN-based framework to build a secured architecture for IoT environment [4]– [5]. However, these researches are only about designing the framework of components and lack of presentation on a solid mechanism for the controller to detect or prevent attacks. In addition, although SDN is a flexible solution in managing networks through a central device, it is still questionable to use SDN directly since attack detection normally requires a lot of statistical information which can be hardly achieved in the South bound interface of the SDN protocol.

Several studies have proposed to use the advances of edge computing in the field of IoT security [6] due to the above-stated benefits. Security frameworks in [7]– [8] are examples of this. However, these researches only design their frameworks, and do not provide a detection algorithm nor performance evaluation for the designs. Authors in [6] also proposed an edge-centric architecture in contrast to the traditional layered architecture found in [9] and [10]. Our solution follows the latter model as we believe the Cloud centralization is necessary for the sake of application services, big data and model optimization.

Some other studies focus on attack detection algorithms in IoT networks [10]– [11] but to the best of our knowledge, none of them considered reducing the algorithmic complexity for faster system response. One of the effective mechanisms applied in the IoT environment for attack identification is the Intrusion Detection system [12] in which the authors proposed a method to generate the rules for signature-based detection, but the accuracy was not investigated. In common attack detection algorithms, Neural Network (NN) is especially popular. Despite its longer training and processing time compared to other algorithms, its high accuracy [3] and adaptability make it worth considering. Indeed, other researches [10]– [12] have confirmed this statement, incorporating NN in the IoT threat detection. However, we believe it is possible to further improve the processing time of NN by reducing the number of data dimensions as proposed in [10] while still keeping minimal accuracy degradation. In [10], the authors rank the quality of features taken based on statistics and use one-class classification with only the best features instead of all of them to reduce complexity. Their solution is found to have only a minor reduction of accuracy. Moreover, work [10] and [11] generated their own data set, we believe that using a publicly available dataset, such as the BoT-IoT dataset [27], is more preferable, as it makes it easier for future researchers to compare results with each other. In this paper, we propose to use a well-known PCA (Principal component analysis) [13] for feature engineering technique combined with an optimized NN, performing multi-layer classification to detect attacks of different types at the same time. PCA is faster and computationally cheaper than other possible feature processing method, for example Autoencoder.

From another research perspective, in recent years, Federated Learning [14] has emerged as a mechanism for federated training problems, such as IoT security [15], [16] or downstreaming [17]. With Federated Learning, instead of sending data to a centralized cloud server for training, each end user or client instead trains a model with their own data, and only sends that model to the server for aggregation. This way, less communication is required as the model is much more light-weighted than the data, and also the privacy of users' data is preserved.

In the domain of Federated Learning, we can find multiple research directions. The authors of work [18] assume a system that is using Federated Learning for training. In this work, the authors concern the way to protect the federated learning's weight updating process from free-rider clients who fake weights during the updating process. However, Federated Learning in this work is not considered as a security solution to detect sources of attacks at all. So this work is not comparable to our proposal. In the same problem, authors in [19], [20] also consider unreliable clients who can send fake weights that affect the Federated learning itself.

There is also the problem of data leakage. The study [21] presents the danger of a malicious Federated Learning server that sends forged weights to participants, then analyze the plaintext weights that are sent back to expose their data. The authors then propose a weight encryption scheme that help clients individually find out whether the weights they get from the server are legitimate or not. Meanwhile, the authors in [22] remove the centralized server, and instead propose a peer-to-peer federated learning model with blockchain for securing data sharing in industrial IoT. Another group of papers made adjustments to federated learning to better suitable for end devices with limited resources. Work [23] made adjustments to the algorithm, while [24] and [25] propose an incentive mechanism with resource consideration. In addition, paper [15] applies federated learning to solve the security problem for IoT. It uses device-type specific models with GRU for anomaly detection along with federated learning. However, GRU has a very long training time and may cause problem in low-end devices. Our approach has better accuracy and shorter training time.

In this paper, we propose a detection framework for multiple types of attacks using a neural network. The neural network has light weight thanks to the reduction of input dimension, reduction of the number of layers and the number of neurons in the hidden layer. In fact, this reduction decreases the detection complexity while keeping the detection performance high. In our study, the overall architecture is deployed both in the centralized manner (i.e training in the Cloud) and federated manner (i.e training at the edge) which is federated with light updating at the Cloud.

III. PROPOSED SECURITY SYSTEM ARCHITECTURE - LocKedge

A. DESIGN OF EDGE-CLOUD SYSTEM ARCHITECTURE

There have been a variety of IoT architectures [1] depending on the function required by different fields. The authors in [4], [9] proposed five-layers architecture for IoT networks as shown in the left side of Fig.1. [9] proposes an architecture including Business layer, Application layer, Processing layer, Transport layer, and Perception layer. While [4] proposes an architecture of Business layer, Application layer, Middle ware layer, Network layer, and Perception layer. In our architecture, we propose to have an additional layer: Edge layer which could distribute computing tasks better, that in turn helps to detect attacks near the source faster, as shown in the right side of Fig.1.

Our Edge-Cloud security architecture is designed to: (1) have low complexity in analyzing data, (2) be capable of detecting early attack right at edge zones and (3) have accurate attack detection with high reliability. With all these goals, the system not only avoids having been badly damaged before successfully detecting attacks but also adapts quickly to the development trend of IoT network in the future with security and scalability requirements.

Detailed descriptions of the layers are as follows:

- *Data perception layer*: IoT devices with sensors.
- *Edge layer*: consists of IoT Gateways which support wired or wireless network access protocols such as Bluetooth, Wi-Fi, 6LoWPAN, NFC, Wi-Fi Direct, 4G-LTE, Lo-Ra, NB-IoT, and so on. An IoT Gateway is responsible for normalizing their data before performing a multi-attack detection. At the Gateway, we develop an accurate light-weighted multi-attack detection module called LocKedge that can detect different types of attacks. When an attack is detected, the Gateway traces its source then blocks the malicious connections. The Gateway can either send its processed data to the Cloud for data mining purposes in the centralized mode (i.e. centralized learning), or train the detection module locally and then sends the weights of the model to the Cloud for aggregation in the federated mode (federated learning). In case of emergency when a particular attack is too intense or its source cannot be determined in time, the Gateway can simply block all incoming data from its zone, not affecting the Cloud or any other legitimate sources of other zones. Detecting and mitigating attacks right at each zone will make the system response faster and more effectively since: (1) it is near the attack sources so detection time is smaller; (2) it has to deal with a smaller set of data from one zone only and thus lessen the processing time and computing capacity requirement; and (3) in the worst case scenario, only the affected zone is down, the Cloud is still protected and the damage is minimized.
- *Network layer*: The network layer which secures data transfer from the lower layer to the higher layer, so it plays an important role in a general architecture.

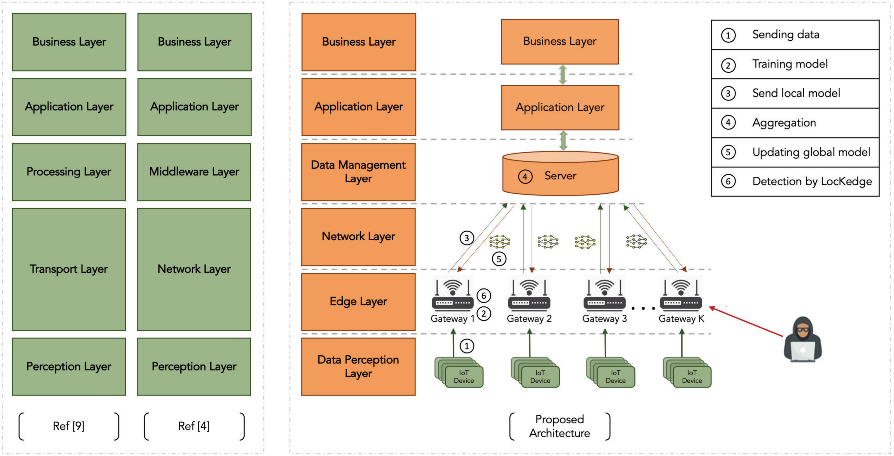

FIGURE 1. **Our architecture vs. 2 reference architectures.**

- *Data management layer*: the Cloud. Within the scope of security, the Cloud is in charge of analyzing given IoT-device's data sets. The Optimization Module developed in the Cloud is responsible for analyzing data, and deciding the number of neurons per layer as well as the weights of the NN algorithm. Periodically, the Cloud sends the aforementioned information to all gateways. Hence, the Cloud deals with big data and the computing phase, and its new rules will be updated to all IoT gateways for more efficient protection of the network.
- *Application layer*: is responsible for inclusive applications management based on the processed information in the Data management layer such as intelligent transportation,smart car, smart health, identity authentication,smart glasses, location, and safety, etc. This layer is providing all kinds of applications for each industry.
- *Business layer*: Business layer functions cover the whole IoT applications and services management. It can create business models, flow charts, executive reports, so on base on data received from lower layers and effective data analysis process. It will help the functional managers or executives to make more accurate decisions about the business strategies base on analysis results

B. DATA PRE-PROCESSING AT THE EDGE

Before the detection phase in which a detection algorithm only takes numerical input, raw traffic needs to be normalized since the data is both categorical and numerical, with numerical data being in vastly different ranges. First, categorical data will be converted to numerical data. Then, all data will be transformed into values in the range between 0 and 1 through the min-max normalization method as follows:

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{1}$$

where: x_i, z_i $(i = 1, 2 \ldots d)$ are values before and after normalization of one data feature and d is the dimension of data.

In fact, in this architecture, we will develop 2 learning modes: (1) a centralized-learning based, (2) a federated learning-based. Our design contribution can be summarized as follows:

- Feature extraction is analyzed in the Cloud to define which features are important to use for detection. It helps the system to reduce complexity for computing full features of a dataset.
- Centralized learning in the Cloud to define the number of neurons per hidden layer and the number of layers in order to receive high accuracy and low complexity of the detection phase.
- Design and evaluate the centralized and federated-learning based detection solutions to cope with facts of IoT networks

In the following section, we will present the detection solutions in both ways aforementioned. We will also elaborate the reasons to use each of the solutions and perform an evaluation for the two methods.

IV. DETECTION MECHANISM

Basically, our detection mechanism is based on a feature extraction module and a classification module as shown in Fig. 2. The feature extraction phase aims at reducing the number of features of incoming samples that are fed to the detection phase. Reducing the dimension of data for detection algorithms is always critical, especially if the data analytic is carried out at the edge devices with low computational capacity and energy supply [3]. This extraction phase also increases the efficiency of the detection phase and reduces the time taken for a system to respond and record information. The detection module is implemented with a NN performing multi-class classification to detect different types of attacks at the same time. Again, we propose to optimize *NN* in terms

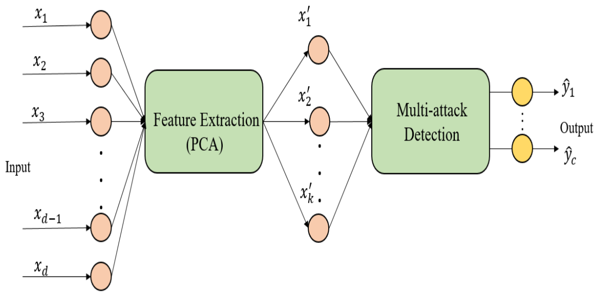

FIGURE 2. Multi-attack detection mechanism.

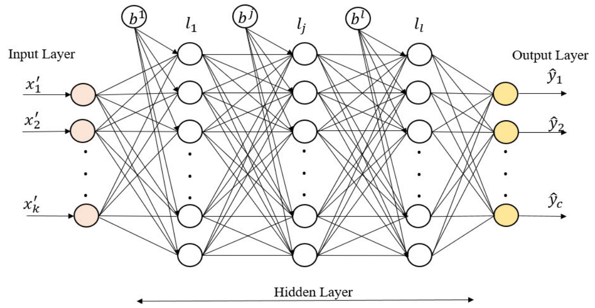

FIGURE 3. Neural network architecture.

of the number of layers and the number of neurons in the hidden layer to minimize the algorithm's complexity while still ensuring high detection accuracy.

According to Fig. 2, normalized data with d dimension will be passed through the Feature Extraction module to perform feature extraction and the dimension of data will be reduced to k features ($k < d$). Those k features will be then used to perform multi - class classification by the NN module afterward.

A. FEATURE EXTRACTION AND DIMENSION REDUCTION MODULE

There are many techniques and corresponding algorithms that can reduce the dimensions of data, the authors in [26] have divided them into 4 groups: Feature Ranker, Feature Evaluator, Dimensionality Reduction and Clustering Algorithms. In this research we experimented with some different algorithms with the BoT-IoT dataset [27] to evaluate the performance. Finally, we choose the Principal Components Analysis (PCA) method [13] to extract the most important features due to its better performance and since PCA is fast and computationally cheap. With PCA, the original data points will be transformed into a new space, where it is possible to differentiate the importance of the components together. The size of data dimensions is decreased from d to k which are k important components of the data in the new space system.

Let's consider the input data matrix including N row vectors $\mathbf{X} = \{x_i\}(i = 1 \dots N)$ where $x_i = \{x_{i1}, x_{i2}, x_{i3} \dots x_{id}\}$ with d is the original dimension of the data. To extract the principal components of $\mathbf{X}$, we calculate the empirical mean of $\mathbf{X}$: $\bar{x} = N^{-1} \sum_{i=1}^{N} x_i$ and the mean-centered matrix $\mathbf{M}$. Each row vector of $\mathbf{M}$ is given as $m_i = x_i - \bar{x}$. Then, we compute the eigenvalue decomposition of the covariance matrix $V = N^{-1} M^T M$ to get the principal components. The relationship between eigenvalues λ and eigenvectors U of square matrix V satisfies equation (2)

$$V\lambda = \lambda U \tag{2}$$

In which, λ is a diagonal matrix, each value λ_i is the i_{th} eigenvalue corresponding eigenvector u_i of matrix U. The eigen decomposition of V is given by:

$$V = \lambda U \lambda^{-1} \tag{3}$$

The principal components of matrix $\mathbf{X}$ are the first k vectors of $\mathbf{V}$ that correspond to k largest eigenvalues. $V_k = \{v_1, v_2, \dots, v_k\}$, which form a subspace close to the distribution of the normalized data. To choose k, we can rely on the amount of information retained in the new data point by selecting the first k values of the eigenvalue that capture 90% or 95% of the sum of the eigenvalues. In our experiment, $k = 9$ is found to ensure capturing over 95% of the total sum of the eigenvalues. New data with the reduced dimensions is the coordinates of the data projected on the new space.

$$X' = MV_k^T \tag{4}$$

B. MULTI-ATTACK DETECTION MODULE

In this module, we deploy a NN to detect multiple types of attacks as shown in Fig. 3. Outputs of the NN are defined as different attack types. Optimizing the number of layers and the number of neurons per layer is directly related to the algorithm complexity. Therefore, in this study, we try to optimize the number of layers and neurons per layer for the BoT-IoT data set to balance complexity and accuracy performance for the multi-attack classification problem.

Fig. 3 illustrates the main components of a NN. The function of NN is to perform complex mapping and convert input information into outputs, which is defined mathematically as $F : R^k \rightarrow R^m$.

The network input is $x'_i = \{x'_{i1}, x'_{i2} \dots x'_{ik}\}$ where k is the dimension of data after being processed by the Feature extraction and Dimension reduction Module and W^j is the weight value of j^{th} layer ($j = 1 \dots l$) with l being the number of hidden layers. The output of each hidden layer is obtained by adding the bias with the products of each input and its corresponding weight W^j, then applying the activation function f, as shown in (5) (define $S_0 = X$)

$$S_j = f(W^j S_{j-1}^T + b^j) \tag{5}$$

Non - linear activation function is required to make a NN to work in case of complicated data such as videos, images, speech... In our model, we choose the activation function in hidden layers is ReLU for higher performance and faster convergence [28] and softmax activation function for the Multi-class classification problem. Their mathematical formulae are shown as below:

$$ReLUf(x) = \begin{cases} 0 & x \leq 0, \\ \text{x} & x \geq 0 \end{cases} \quad Softmax\sigma(z_i) = \frac{e^{z_j}}{\sum_{j=1}^{k} e^{z_j}} \tag{6}$$

Accordingly, the input information X passes through each layer and is transformed by (5) until it reaches the output layer with the softmax activation function for the multi-class problem, which is denoted as $\hat{y} = softmax(W^{out}S_l^T)$. This process is called Forward propagation.

C. LocKedge OPERATION

As mentioned above, the detection module is implemented at the Edge in the IoT system to achieve faster detection near attack sources thereby enabling quicker system response. Detection at the edge also allows us to treat traffic within an narrowed attacked zone locally without affecting other edge zones with our security policies.

From the NN's performance perspective, we propose to optimize a parameter for the training process. The Loss Function, calculating the difference between the predicted value $\hat{y}$ and the actual value y will be used to adjust the training process and learning efficiency of the NN, so that the model can best fit with the data used. In our design, we use the cross-entropy loss function (7) for multi-class classification with c classes for c types of attack in the Bot-IoT data set such as DoS – TCP, DDoS – TCP, OS – Fingerprinting,…

$$L(\hat{y}, y) = -\sum_{i=1}^{c} y_i log(\hat{y}_i) \tag{7}$$

In the training process, the network is trained by a set of labeled data (x_i, y_i), which is used to reduce the average loss value after each iteration. To get better learning efficiency by adjusting weight (w) and bias (b) in the NN, the loss function (8) needs to be minimized.

$$J_{(w,b)} = \frac{1}{N}\sum_{=1}^{N} L(\hat{y}(x'_i), y(x'_i)) \tag{8}$$

In LocKedge, the structure of NN (including the number of hidden layers and neurons per hidden layer) is designed at the beginning and will not be changed with the complexity of the system and the model at the edge can be updated in either the centralized or the federated learning manners. In both scenarios, the subspace matrix V_k (3) of PCA is calculated in the cloud, based on the previously archived data, V_k is then sent to the edge along with the trained detection model in the centralized mode or with the initialized model in the federated mode. Each of the manners brings specific pros and cons as we will describe in the next subsections.

1) CENTRALIZED LEARNING MODE

For centralized learning, all raw data is sent directly to the cloud, then pre-processed for the training process, and hereby this data is also stored. The training process is done locally in the cloud and finished when the loss function converges. At this time, the NN parameters have been optimized and sent to the edge devices to update the new model. This way, the Cloud has an overview of the overall system including multiple different edges. Hence, the detection is potentially accurate. However, it is obvious that it will make a big burden to the Cloud to compute for a huge bundle of data sent from the edge.

In fact, there are many techniques existing to optimize the loss function. Reference [29] proposes an algorithm to implement this optimization function. To fasten the convergence in a deep NN-based model, we should use an adaptive learning rate algorithm. In the centralized mode, optimized by Adam's algorithm, the formula (9) shows the rule to calculate and update the parameter of Adam's method described in [30].

$$w' = w - \alpha\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}; b' = b - \alpha\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{9}$$

where: α is learning rate, $\hat{m}_t$ and $\hat{v}_t$ is the bias-corrected first (the mean) and the second (the uncentered variance) moment estimates and ϵ is a smoothing term that avoids division by zero.

These values will be updated after each iteration, until the value of loss the function reaches a minimum. And this process is called backpropagation.

The pseudocode for Centralized Learning mode is stated in Algorithm 1.

Algorithm 1: Centralized Learning LocKedge

```
Model training stage: Computation at the Cloud: Run
PCA on archived data to get subspace matrix $\mathbf{V_k}$
Initialize model weights $\mathbf{w_0}$
Collect data $\mathbf{X_{N\times d}}$ from all Edge gateways
Reduce data dimension to $k$ using Equation 4
for $t = 1$ to $e$ do
    for each data batch do
        Perform Forward Propagation to calculate loss
          as in Equation 8
        Perform Backpropagation to update weight
          values with Adam Optimization as in
          Equation 9
    end
end
Send trained model weights $\mathbf{w_e}$ to all gateways
 Detection stage:
Computation at the Edge:
Use trained model to detect attack
Return label $\mathcal{L}$ of new data
```

2) FEDERATED LEARNING MODE

Due to the distributed nature of the IoT ecosystem and the unreliability of wireless transmission, sending all user data to the Cloud for model training may be costly and time consuming. Furthermore, this approach will also have the risk of exposing private or sensitive user data. All of this can be solved by doing the detection as well as the training phase in the edge. However, as each edge only has access to its own data, which is often small and limited, and the data between the edges can be very different, the quality of the resulting edge-trained model may not be good enough. Using Federated Learning, this problem can be mitigated as the edges can "communicate" with each other through the aggregated weights of the server, while still avoiding sending data directly, saving bandwidth as well as protecting privacy.

Federated Learning is a distributed machine learning technique, in which the training process of a model as well as the data involved is divided between multiple parties - or "clients", and no client has access to the data of another. Instead, clients only send the weight information of the local model they train with their own data, either directly with each other in their peer-to-peer model or through a centralized aggregator server in a client-server model. In this paper, we opt for the centralized model, as it has a faster convergence time, as well as having a better fit for our architecture.

In our client-server model, the server firstly decides the Feature Extraction phase, as well as the hyper-parameters and the initial weights of the NN model, then sends this information to all clients. Then, each client will train its model with its own data, using Stochastic Gradient Descent for E local epochs. Afterwards, all clients will send the updated weight of their model to the server, which will then calculate the aggregated weight using the following formula [14]:

$$w_t = \sum_{k=1}^{K} \frac{n_k}{n} w_t^k \tag{10}$$

where K is the number of participating clients. The server will then send the calculated weights for all clients to update their model with, completing one communication round. Repeating this process for C communication rounds (with C sufficiently large) and all clients will end up with a well-trained model which is generalized for all the local data, with no data transmission required.

In the federated learning mode, we use the traditional Stochastic Gradient Descent optimizer, which can be written as:

$$w' = w - \eta \nabla_w J_{(w,b)};\ b' = b - \eta \nabla_b J_{(w,b)}; \tag{11}$$

with the learning rate $\eta = 0.01$.

The pseudocode for the Federated Learning mode is stated in Algorithm 2.

Algorithm 2: Federated Learning LocKedge

```
Model training stage:
Computation at the Cloud:
Run PCA on archived data to get subspace matrix $\mathbf{V_k}$
Initialize model weights $\mathbf{w_0}$
Send initialized model and subspace matrix to all
  gateways
Each gateway reduce its data dimension to $k$ using
  Equation 4
for each communication round $T = 1$ to $C$ do
    for each gateway $k \in \{1, \ldots, K\}$ in parallel do
        ClientUpdate($k$, $T$)
        Send updated weights $\mathbf{w_T^k}$ to server
    end
    Aggregate weights using Equation 10
    Send aggregated weights $\mathbf{w_T}$ to gateways
end
Detection stage:
Computation at the Edge:
Use trained model to detect attack
Return label $\mathcal{L}$ of new data
function ClientUpdate(k, T):
    Initialize weights of local model with $\mathbf{w_{T-1}}$ received
      form server
    for each local epoch $i = 1$ to $E$ do
        for each data batch do
            Perform Forward Propagation to calculate
              loss as in Equation 8
            Update the weight value with Stochastic
              Gradient Descent as in Equation 11
        end
    end
end
```

V. PERFORMANCE EVALUATION

In this research, we use the BoT-IoT data set [27] to evaluate our model. This data set was generated by designing a realistic IoT network environment, with five IoT scenarios: a weather station, a smart fridge, remotely activated, motion activated lights and a smart thermostat [31]. We used version 5% extracted from the original data set proposed in [31]. It includes 10 types of attacks: DDoS (HTTP, TCP, UDP), DoS (HTTP, TCP, UDP), OS Fingerprinting, Server Scanning, Keylogging and Data exfiltration attacks. We use roughly 70 percent of data for training and the remaining 30 percent for testing. The quantity of each attack type is shown in Table 1.

In our test scenario, we will evaluate LocKedge in both manners of centralized based learning and Federated learning (i.e. distributed). In fact, in terms of complexity (i.e processing steps), centralized learning and federated learning are similar because they process the same tasks which can be assigned to the edge or cloud.

A. COMPLEXITY EVALUATION

In LocKedge, the first phase is the feature extraction done by PCA, with given matrices $X, M \in R^{N \times d}$; $V \in R^{d \times d}$. Two most computationally intensive tasks are multiplying two matrices with the size of $N \times d$ and $d \times N$ (i.e computing the covariance matrix), which has the complexity of $O(Nd \times min(N, d))$, and calculating the eigenvalue decomposition.

TABLE 1. Statistics of the BoT-IoT dataset.

Types of Attack	Number of samples
DoS-HTTP	1485
DoS-TCP	615800
DoS-UDP	1032975
DDoS-HTTP	989
DDoS-TCP	977380
DDoS-UDP	948255
OS Fingerprinting	17914
Server Scanning	73168
Keylogging	73
Data Theft	6
Normal	477
Totals	3668522

Given in [32], the computational complexity for eigenvalue decomposition with square matrix size $d \times d$ is $O(d^3)$. Thus, the time complexity of the PCA algorithm is given by:

$$O(Nd \times min(N, d) + d^3) \tag{12}$$

And then, that data with the reduced dimension k is fed to a NN. In this model, we use only one hidden layer on the NN structure to optimize the complexity. NN with three layers is enough to represent an arbitrary function according to the [33]. Let h be the number of neurons in the hidden layer and c is the number of outputs or the number of classes. For the forward pass from the input layer to the hidden layer, like in equation (5), we have $S_{hN} = W_{hk} \times X^T_{Nk}$, where W_{hk} is the weight matrix with h rows and k columns. The time complexity for this matrix multiplication is $O(hkN)$, and $O(hN)$ is the complexity of applying the activation function. The total complexity of this two-step is, therefore, given by:

$$O(hkN + hN) = O(hN(k + 1)) = O(hNk)$$

Similarly, from the hidden layer to the output layer, the complexity is given by $O(ckN)$. so, in total, the time complexity for the forward propagation process at each iteration is:

$$O(hNk + chN) = O(N(kh + hc)) \tag{13}$$

The backward propagation starts from the output layer to the hidden layer by backward propagating the error matrix L_{cN}. Adjusting the weight matrix between these layers, we have $W'_{ch} = W_{ch} - L_{cN} \times S^T_{hN}$, which has the time complexity of $O(cNh)$. Similarly, backpropagating from the hidden layer to the input layer has the complexity of $O(hkN)$. In total, for one iteration, the time complexity of the backward propagation process is $O(N(ch + hk))$, which is the same as Formula (13). Thus, we can determine the total time complexity of the forward and backward propagation for one epoch is $O(N(kh + hc))$. If we train the NN model with e epochs, the total complexity is given by:

$$O(eN(kh + hc)) \tag{14}$$

So, for a generic NN with many hidden layers in which h_i is the number of neurons in hidden layer i, we can determine the time complexity by the formula:

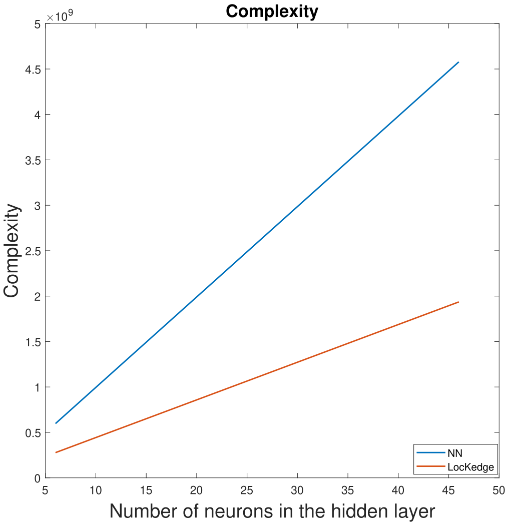

FIGURE 4. The complexity of LocKedge vs. NN.

$$O(eN(dh_1 + \sum_{i=1}^{l-1} h_i h_{i+1} + h_{l-1}c)) \tag{15}$$

In the worst case, the total time complexity of LocKedge is:

$$O(Nd \times min(N, d) + d^3 + eN(kh + hc)) \tag{16}$$

To achieve the goal of reducing the time complexity compared to the original NN, the following condition must be met: $O(Nd \times min(N, d) + d^3 + eN(kh + hc)) < eN(dh + hc)$ From then, it can be deduced that k must be chosen so that:

$$k < d(1 - \frac{min(N, d)}{eh} - \frac{d^2}{eNh}) \tag{17}$$

In reality, security data sets often have input data dimension that is much smaller than the number of samples, and thus, $min(N, d) = d$. As h gets bigger, complexity will also increase. With the condition $d \ll N$ and the number of epochs e is big enough for d, then $\frac{d}{eh} - \frac{d^2}{eNh}$ in (17) will be close to 0. Thus, (17) becomes $k < d$, in other words, the time complexity of LocKedge will always be better than a conventional NN. In general, we can consider the LocKedge architecture to function as a NN with a better complexity.

Fig. 4 shows how the complexity is obtained by Equation (16) when the number of neurons of the hidden layer (i.e. h) varies from 6 to 45 in our test scenario. In Fig. 4, we can see more neurons in the hidden layer. The higher complexity of the algorithms, the much faster rate the NN's complexity increases. In fact, the complexity of NN is always about 2 times higher than our architecture's complexity. Therefore, this architecture provides better efficiency in optimizing the complexity than that of the traditional multi-layer NN.

In this study, we also measure the training time performance which is calculated by the time of reading data in for the training phase till the point the training process ends, returning a new model. The training time is presented

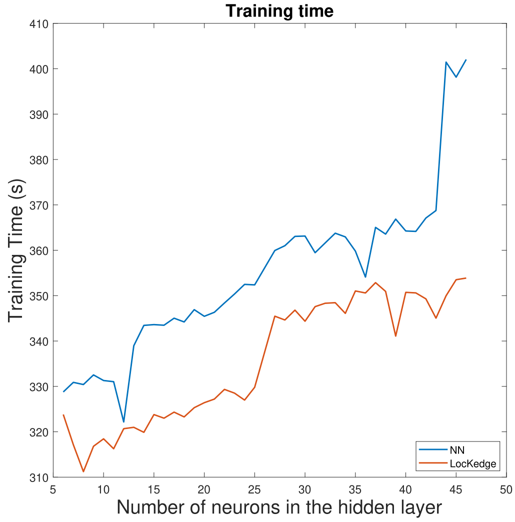

FIGURE 5. Training time.

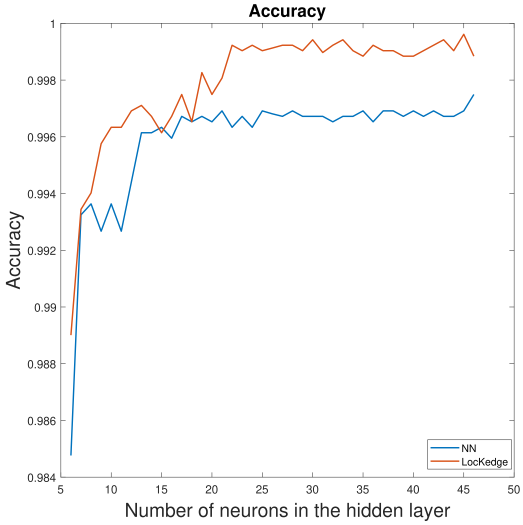

FIGURE 6. Accuracy comparison between NN and LocKedge.

in Fig. 5. As shown in Fig. 5, the training time of LocKedge is lower than NN in the context of the BoT-IoT data set. This is entirely consistent with the theory mentioned above. With remarkable points in terms of time or low complexity, the LocKedge mechanism is likely to be suitable for the distributed models on edge devices where processing capacity is more limited than that of the centralized architecture.

B. DETECTION PERFORMANCE OF THE CENTRALIZED-LEARNING LocKedge

In the centralized scenario, all the data is gathered at the Cloud server for training and testing of the model. Thus, for this evaluation, the dataset was used in one piece, without being divided into smaller datasets, as is the case in the next section.

Firstly, we compared the performance of the centralized LocKedge with the pure NN Model (NN) without the feature processing in terms of Accuracy, Detection Rate (DR) and Complexity when performing a multi - attack classification. The number of neurons in the hidden layer h in our experiment ranges from 6 to 46. The results are shown in Fig. 6, where:

- Accuracy is the total number of correctly predicted samples in all tests.
- Detection Rate (DR) is the number of the actual positives that are predicted as positive.

In Fig. 6, we can see the Accuracy of the centralized LocKedge is higher than NN. The accuracy remains stable at about 0.999 for the centralized LocKedge while NN only reaches about 0.997. Also, we can see that the accuracy tends to increase and stabilize as the number of neurons in the hidden layer increases. With the results shown in Fig. 6, the accuracy of the centralized LocKedge is stable when $h = 22$ neurons and with NN is $h = 17$ neurons. This shows that we do not need to use too many neurons in the hidden layer to obtain optimal accuracy.

TABLE 2. Detection rate of LocKedge vs DNN, RNN, CNN, NN.

Attack type	DNN	RNN	CNN	NN	LocKedge
DoS-HTTP	0.96699	0.96868	0.97512	0.76091	0.90862
DoS-TCP	0.96628	0.96772	0.97112	1	1
DoS-UDP	0.96525	0.96761	0.97112	0.99928	0.99928
DDoS-HTTP	0.96616	0.96564	0.97010	0.98662	0.98715
DDoS-TCP	0.96219	0.96650	0.97003	0.99941	0.99965
DDoS-UDP	0.96118	0.9666	0.97006	0.99946	0.99946
OS Fingerprinting	0.96139	0.96762	0.97001	0.98887	0.99258
Server Scanning	0.96428	0.96874	0.97102	0.99947	0.99973
Keylogging	0.96762	0.96999	0.98102	0.98780	0.99268
Data Theft	1	1	1	0.46341	0.56098

In addition, we evaluate the average detection rate for all values of h for each attack type between LocKedge, NN and other Deep Learning (DL) method: DNN, Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN). These results are taken from the study [34] done on the same Bot-IoT data set. The results are shown in Table 2

Moreover, we compare the centralized LocKedge with some popular Machine Learning algorithms such as K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF) and Support Vector Machine (SVM) shown in Table 3. In general, the average detection rate of the centralized LocKedge is higher than the other methods in most classes, especially superior to the ML methods as seen in Table 3. The centralized LocKedge provides a balanced and more uniform detection rate between classes. However, in terms of DoS-HTTP and Data Theft attacks, the detection rate is lower than of the other methods. It is stemmed from the fact that both of these attacks have a low number of samples, especially Data Theft which has only 6 samples. So that detecting these 2 types of attacks correctly requires much more complex detection models. Thus, having a lower detection rate on two

TABLE 3. Detection rate of LocKedge vs. KNN, DT, RF, SVM.

Attack type	KNN	DT	RF	SVM	LocKedge
DoS-HTTP	0.81690	0.84507	0.76056	0.74647	0.90862
DoS-TCP	1	0.99752	1	1	1
DoS-UDP	0.99851	0.99926	0.99926	0.99554	0.99928
DDoS-HTTP	0.96774	0.82258	0.96774	0.97581	0.98715
DDoS-TCP	0.99173	0.97746	0.99248	0.99624	0.99965
DDoS-UDP	0.99217	1	1	0.96784	0.99946
OS Fingerprinting	0.93478	0.93478	0.89130	0.78261	0.99258
Server Scanning	0.97826	1	1	0.98913	0.99973
Keylogging	1	0.3	0.9	1	0.99268
Data Theft	0	0	0	0	0.56098

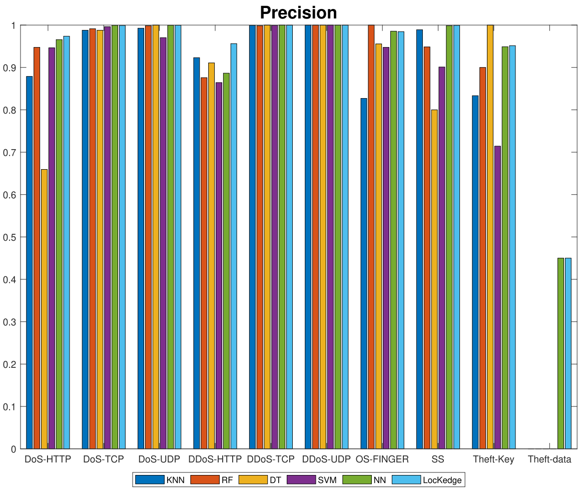

FIGURE 7. Precision of LocKedge vs. other solutions.

types of attack out of ten types is an acceptable trade-off for a lightweight model capable of running on edge devices.

Fig. 7 and Fig. 8 present the overall performance of the centralized LocKedge and some popular Machine Learning algorithms in terms of Precision and F1-score in the classification of multi-attacks detection. Where:

- Precision $= \frac{true\_positive}{true\_positive+false\_positive}$
- Recall $= \frac{true\_positive}{true\_positive+false\_negative}$
- F1-score $= 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$

As we can see, the precision of the centralized LocKedge are always highest compared to other solutions for each type of attacks. Hence it means that our proposed solution can minimize the false positives compared to the mentioned approaches. In terms of F1-score, we also observe that F1-scores of LocKedge are better than of other solutions in most of attack types. It proves to be a good scheme for taking both false positives and false negatives into account.

Finally, we examine the multiclass micro-averaging and macro-averaging ROC Curves for LocKedge in the centralized mode as illustrated in Fig.9. Micro-averaging is plotted by treating each element of the label indicator matrix as a binary prediction, while macro-averaging simply gives equal weight to the classification of each label. We can see that the AUC values in both cases are equal to 1, which is the most ideal result. This shows that the centralized mode LocKedge has uniformity in classifying different labels, a conclusion supported by other measures such as Precision and F1-score.

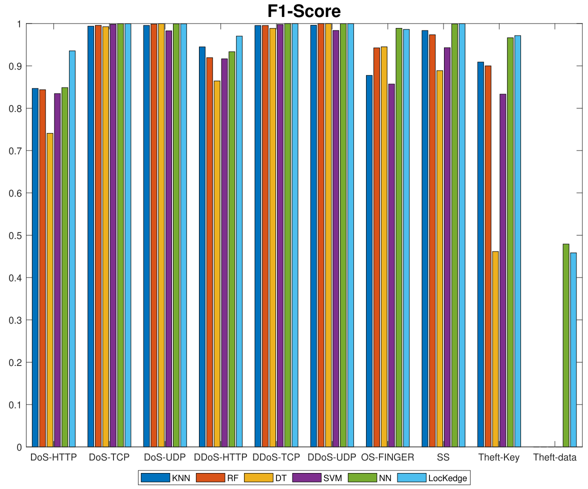

FIGURE 8. F1-Score of LocKedge vs. other solutions.

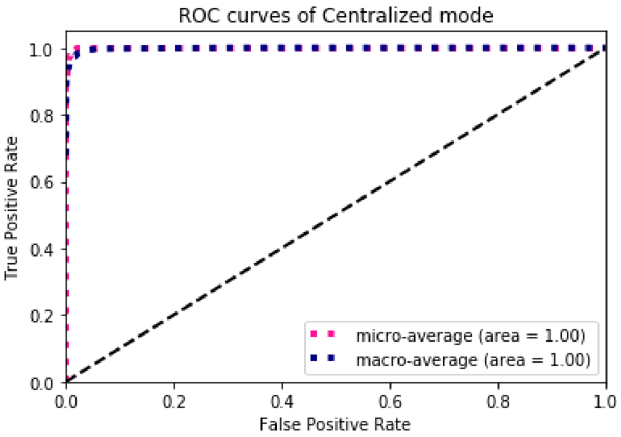

FIGURE 9. Micro-averaged ROC curve in centralized mode.

C. DETECTION PERFORMANCE OF FEDERATED LEARNING LocKedge

As stated before, Federated Learning helps to cope with the fact that detection and training can be done at the edge, near the attack source so attack detection can be more quickly detected and attack sources are more localized. However, training at the Edge with a small set of local data may result in lower performance in abnormal detection. Therefore, in this subsection, we will study the detection performance of the Federated Learning if it can be acceptable in trade off for its own benefits in an IoT network environment.

In our test scenario, we divide the BoT-IoT dataset [31] into four smaller client datasets according to the source IP address in order to simulate an IoT network with 4 different zones where data from clients are sent to four IoT gateways.

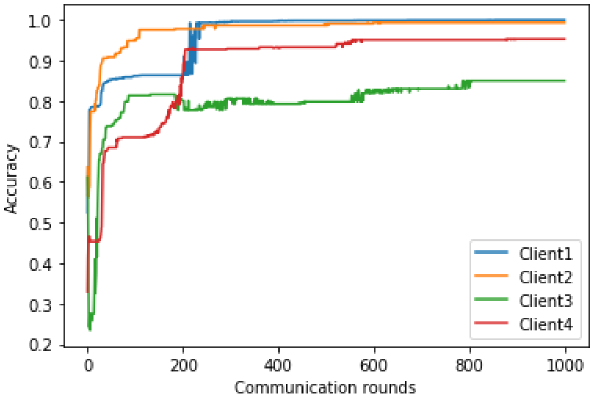

FIGURE 10. Accuracy of test sets, 1000 communication rounds.

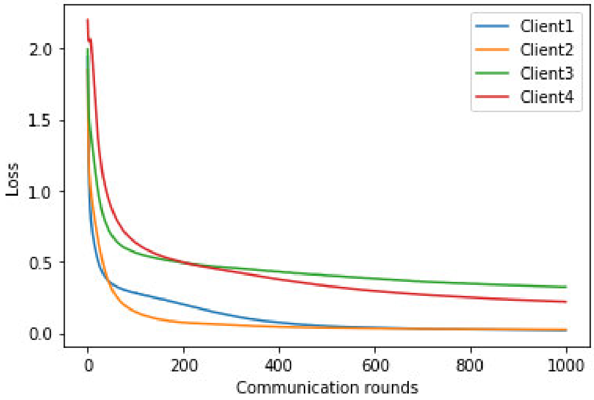

FIGURE 11. Loss values on test sets, 1000 communication rounds.

There are four attacking sources in the BoT-IoT testbed, with their IP addresses ranging from 192.168.100.147 to 192.168.100.150, so we assume that each source attacks a different gateway. All other source IP addresses are treated as normal or victim devices in one of the four zones.

Each dataset is then divided into a training and testing set, so we will have 4 train sets and 4 test sets. The feature extraction phase (PCA) is performed using all 4 training sets together, then the detection model will be trained by the training sets separately using the federated learning approach. After each communication round, the resulting global model will be evaluated using the 4 different test sets.

Fig. 10 and Fig. 11 show the accuracy and the loss after 1000 communication rounds of the test. The number of local epochs was set to 1, after empirical testing showed that this greatly reduces the training time.

As we can see, the accuracy stops increasing and the loss stops decreasing much after about 350 communication rounds, so this can be a good cut-off point. The accuracy also reaches close to 100% for Client 1,2 over 90% for Client 4, and about 80% for Client 3; comparable to that of the centralized approach.

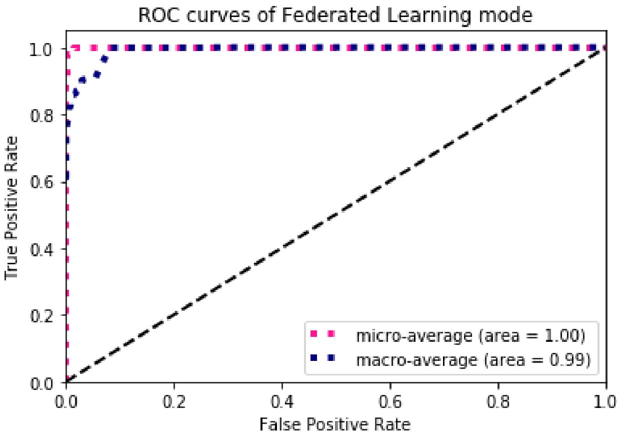

FIGURE 12. Averaged ROC curve in the federated learning mode.

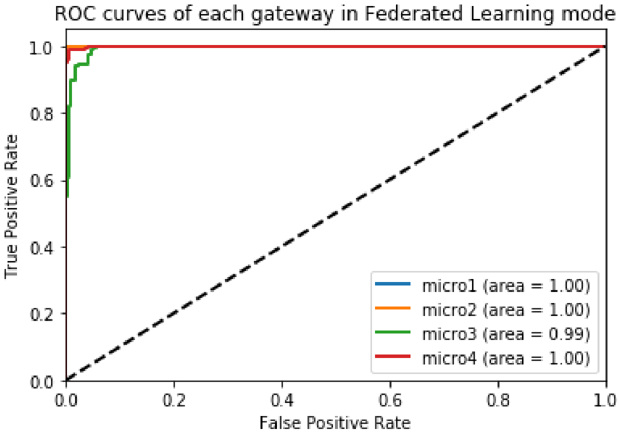

FIGURE 13. ROC curve for each edge gateway in the federated learning mode.

The ROC curves of the Federated mode are presented in Fig.12 and Fig.13. In Fig.12, the macro-averaging AUC (Area Under the ROC Curve) is a bit lower when compared to the centralized mode. This can be explained by the different distribution of the data at different nodes, as well as the fact that some nodes may not have all the labels. The micro-averaging evaluation results for each node are shown in Fig.13. Save for node 3 with the AUC of 0.99, all AUC values are equal to 1.

We also performed a small comparison between LocKedge in the centralized and federated learning mode in terms of F1-score, detection rate and precision. The results are shown in Fig. 14, 15 and 16, respectively. We can see that in some types of attacks such as DoS-HTTP, DDoS-HTTP and theft-data, the result in federated learning mode is inferior to its centralized mode counterpart, but remains acceptable (greater than 65%). This can be explained that due to the uneven data distribution among clients, the number of samples with these types of attacks may be small or none at all in some nodes, which will affect the training process. In practice, with the

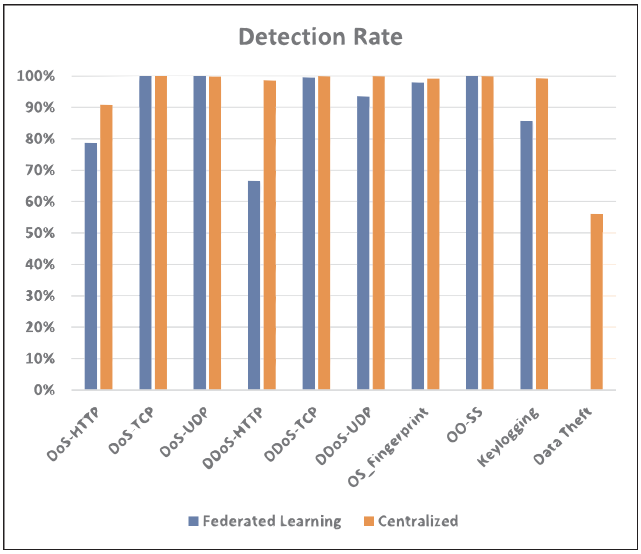

FIGURE 14. Compare detection rate of FL and CL mode.

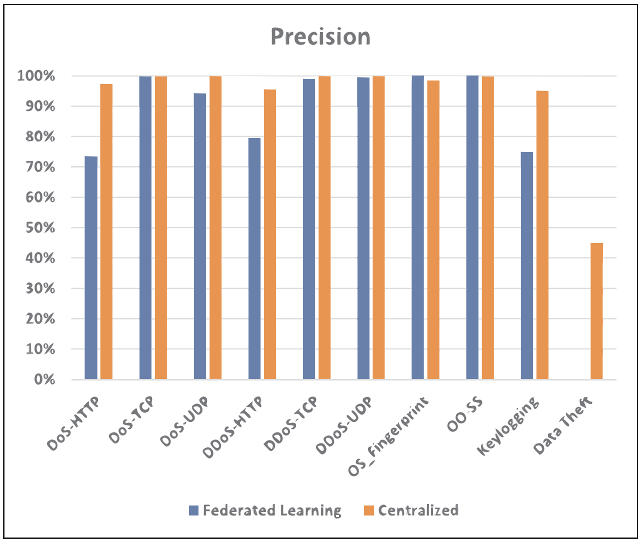

FIGURE 15. Compare precision of FL and CL mode.

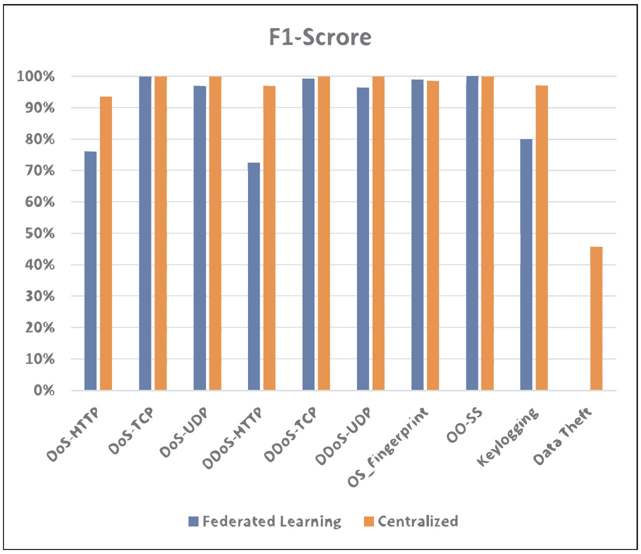

FIGURE 16. Compare F1-Score FL vs CL mode.

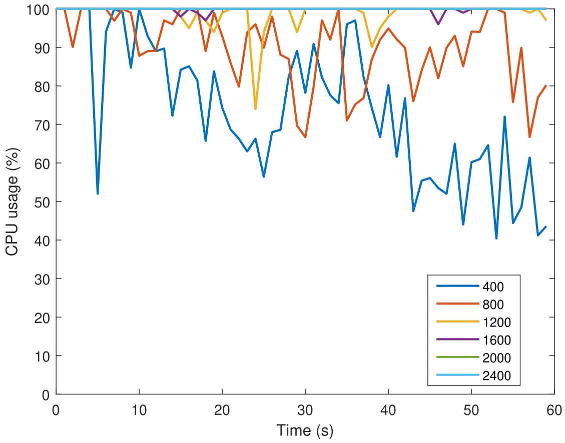

FIGURE 17. CPU usage of a core under rate of 400 to 2400 samples per second.

federated learning mode, this is quite normal since different clients will have their own source of data, and thus some zone may not have enough labels.

D. EVALUATION OF EDGE COMPUTING CAPACITY

To evaluate the Edge-Cloud architecture, we measure the CPU and RAM usage of the Edge smart gateway which deploys the detection module of LocKedge in different attack volumes. This evaluation helps us to have more insight into computing performance of the edge that shares the computing task with the Cloud distributively.

1) EXPERIMENT SETTING

In our testbed, we use a Raspberry Pi 3B+ to implement the Edge gateway due to its popularity in low power consumption and small size. The the Raspberry Pi 3B+ features a 1.4GHz ARM-based quad-core processor with 400% CPU usage at maximum, and 1GB RAM. The Raspberry Pi OS is installed; and our detection solution is programmed with Python 3.

2) PERFORMANCE EVALUATION

To investigate the CPU usage of the PI3 while deploying the detection module, we load traffic of 400 to 2400 samples per second to the PI3. Fig.17 describes the CPU usage of a core of PI3 under attack rates from 400 to 2400 samples per second. We can see that the rate of 2400 samples per second reaches 100% of the CPU usage of a core among a quad-core.

Over time, the CPU usage goes up and down for each attack rate represented by each line in Fig.17. If we calculate an average of the CPU usage for each line, and then compare the CPU usages of each different rates, then the behavior of the core is represented in Fig.18. It can be seen that the CPU

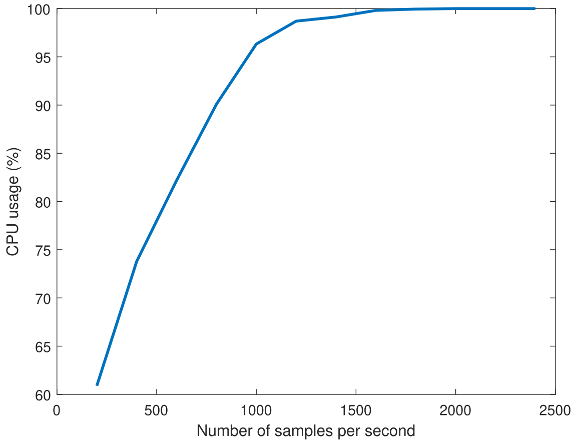

FIGURE 18. CPU usage of a core under rate of 400 to 2400 samples per second.

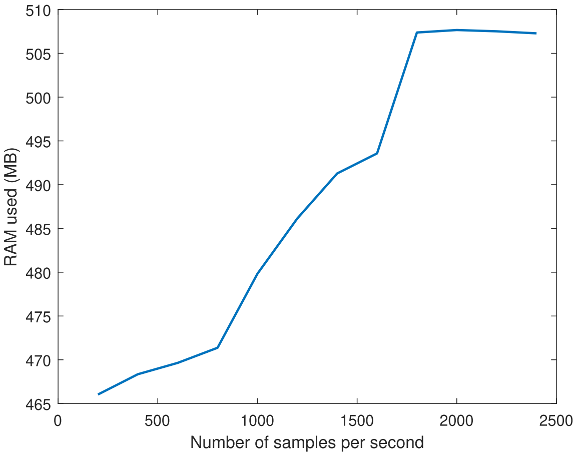

FIGURE 19. RAM usage with different attack rates.

usage increases exponentially as the attack rate increases and get saturated by the attack rate of 2400 samples per second. It can be deduced that for the whole quad-core, the Pi3-based edge can tolerate the attack rate of 9.600 samples per second.

As the attack rates grow, the memory usage of the PI3 also increases as illustrated in Fig.19. In fact, incoming packets must go through a parser which converts information captured from those packets into a new data structure (i.e. samples) that is the input for the NN-based detection module. We observe that, when the sample rate increases up to below the threshold of 1800 samples per second, RAM usage also increases with the whole detection computing at the Edge, as demonstrated in Fig.19. However, from more than 1800 samples per second, the processing speed of the PI3 is not as fast as incoming sample rates. Hence, samples are accumulated gradually at the Parser. Therefore, as long as the sample rate is equal or higher than 1800 samples per second, up to a certain time, the RAM of the PI3 will be consumed all.

In our test scenario, in each edge zone, we measure the sample rates of DDoS-TCP, DDoS-UDP, DoS-TCP, DoS-UDP attack types 3.950, 5086, 1.927, 2.605 samples per second respectively, that are below the attack rate that the PI3 can tolerate during attack detection. Therefore, we can see that it depends how big should an edge zone be organized to distributively monitor traffic of different local areas.For better deploy an edge-cloud architecture, more powerful edge nodes should be chosen to have enough RAM and better clock speed and RAM. For example, Raspberry Pi 4 has a faster 1.5GHz processor, and RAM of 2GB which could work better right at the edge.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have introduced an Edge-Cloud architecture with a low complexity attack detection mechanism – LocKedge, which is suitable for deployment on edge devices. LocKedge can detect multi attacks faster and make use of the resources of the edge layer. The evaluation in terms of complexity, detection rate and accuracy, using real traffic data set ''BoT-IoT'' showed that LocKedge not only decreases the complexity and increases the accuracy but also outperforms the recent machine learning models and deep learning models. It gives a balanced detection between classes for eleven types of attacks. In future work, we will study to improve the detection rate of Theft-Data-typed attacks by getting more data samples and getting some insights into it.

ACKNOWLEDGMENT

The authors would like to thank our students Mr. Nguyen Minh Dan, Mr. Le Anh Quang, and Mr. Le Khanh Nam for contributing to the testbed construction and performance measurement. Also thanks for the support of Hanoi University of Science and Technology, our research in cybersecurity for IoT networks has been continually extended.

REFERENCES

[1] H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, M. Aledhari, and H. Karimipour, ''A survey on Internet of Things security: Requirements, challenges, and solutions,'' *Internet Things*, Nov. 2019, Art. no. 100129, doi: 10.1016/j.iot.2019.100129.

[2] McKinsey. (2020). *Growing Opportunities in the Internet of Things*. Accessed: Jun. 21, 2020. [Online]. Available: http://www.mckinsey.com/industries/privateequity-and-principalinvestors/our-insights/growing-opportunities-in-theinternet-of-things

[3] M. A. Amanullah, R. A. A. Habeeb, F. Nasaruddin, A. Gani, E. Ahmed, A. Nainar, N. Akim, and M. Imran, ''Deep learning and big data technologies for IoT security,'' *Comput. Commun.*, vol. 151, pp. 495–517, Feb. 2020.

[4] S. Kraijak and P. Tuwanut, ''A survey on IoT architectures, protocols, applications, security, privacy, real-world implementation and future trends,'' in *Proc. 11th Int. Conf. Wireless Commun., Netw. Mobile Comput. (WiCOM)*, Shanghai, China, 2015, pp. 1–6, doi: 10.1049/cp.2015.0714.

[5] F. Olivier, G. Carlos, and N. Florent, ''New security architecture for IoT network,'' *Procedia Comput. Sci.*, vol. 52, pp. 1028–1033, Dec. 2015.

[6] K. Sha, T. A. Yang, W. Wei, and S. Davari, ''A survey of edge computing-based designs for IoT security,'' *Digit. Commun. Netw.*, vol. 6, no. 2, pp. 195–202, May 2020, doi: 10.1016/j.dcan.2019.08.006.

[7] K. Sha, R. Errabelly, W. Wei, T. A. Yang, and Z. Wang, ''EdgeSec: Design of an edge layer security service to enhance IoT security,'' in *Proc. IEEE 1st Int. Conf. Fog Edge Comput. (ICFEC)*, Madrid, Spain, May 2017, pp. 81–88, doi: 10.1109/ICFEC.2017.7.

[8] T. Markham and C. Payne, "Security at the network edge: A distributed firewall architecture," in *Proc. DARPA Inf. Survivability Conf. Exposit. II (DISCEX)*, Anaheim, CA, USA, 2001, pp. 279–286, doi: 10.1109/DISCEX.2001.932222.

[9] M. Wu, T.-J. Lu, F.-Y. Ling, J. Sun, and H.-Y. Du, "Research on the architecture of Internet of Things," in *Proc. 3rd Int. Conf. Adv. Comput. Theory Eng. (ICACTE)*, Chengdu, China, Aug. 2010, pp. V5-484-V5-487, doi: 10.1109/ICACTE.2010.5579493.

[10] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer Internet of Things devices," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, San Francisco, CA, USA, May 2018, pp. 29–35, doi: 10.1109/SPW.2018.00013.

[11] J. Canedo and A. Skjellum, "Using machine learning to secure IoT systems," in *Proc. 14th Annu. Conf. Privacy, Secur. Trust (PST)*, Auckland, New Zealand, Dec. 2016, pp. 219–222, doi: 10.1109/PST.2016.7906930.

[12] Y. Soe, Y. Feng, P. Santosa, R. Hartanto, and K. Sakurai, "Rule generation for signature based detection systems of cyber attacks in iot environments," *Bull. Netw., Comput., Syst., Softw.*, vol. 8, no. 2, pp. 93–97, 2019.

[13] I. T. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer, 2002.

[14] H. McMahan, "Federated learning of deep networks using model averaging," 2016, *arXiv:1602.05629*. [Online]. Available: https://arxiv.org/abs/1602.05629

[15] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IoT," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 756–767.

[16] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, "Multi-task network anomaly detection using federated learning," in *Proc. 10th Int. Symp. Inf. Commun. Technol. (SoICT)*. New York, NY, USA: ACM, 2019, pp. 273–279.

[17] R. Fantacci and B. Picano, "Federated learning framework for mobile edge computing networks," *CAAI Trans. Intell. Technol.*, vol. 5, no. 1, pp. 15–21, Mar. 2020.

[18] J. Lin, M. Du, and J. Liu, "Free-riders in federated learning: Attacks and defenses," 2019, *arXiv:1911.12560*. [Online]. Available: http://arxiv.org/abs/1911.12560

[19] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, "Abnormal client behavior detection in federated learning," 2019, *arXiv:1910.09933*. [Online]. Available: http://arxiv.org/abs/1910.09933

[20] T. D. Nguyen, P. Rieger, M. Miettinen, and A.-R. Sadeghi, "Poisoning attacks on federated learning-based IoT intrusion detection system," in *Proc. Workshop Decentralized IoT Syst. Secur. (DISS)*, 2020, pp. 1–7.

[21] A. Fu, X. Zhang, N. Xiong, Y. Gao, and H. Wang, "VFL: A verifiable federated learning with privacy-preserving for big data in industrial IoT," 2020, *arXiv:2007.13585*. [Online]. Available: http://arxiv.org/abs/2007.13585

[22] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.

[23] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[24] R. Zeng, S. Zhang, J. Wang, and X. Chu, "FMore: An incentive scheme of multi-dimensional auction for federated learning in MEC," 2020, *arXiv:2002.09699*. [Online]. Available: http://arxiv.org/abs/2002.09699

[25] L. U. Khan, S. R. Pandey, N. H. Tran, W. Saad, Z. Han, M. N. H. Nguyen, and C. S. Hong, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 88–93, Oct. 2020, doi: 10.1109/MCOM.001.1900649.

[26] N. Sharma and K. Saroha, "Study of dimension reduction methodologies in data mining," in *Proc. Int. Conf. Comput., Commun. Autom.*, Noida, India, May 2015, pp. 133–137, doi: 10.1109/CCAA.2015.7148359.

[27] N. Moustafa. (2020). *The Bot-IoT Dataset*. Accessed: Feb. 25, 2020. [Online]. Available: http://dx.doi.org/10.21227/r7v2-x988

[28] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, vol. 15, 2010, pp. 315–323.

[29] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*. [Online]. Available: http://arxiv.org/abs/1609.04747

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[31] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput.Syst.*, vol. 100, pp. 779–796, 2018.

[32] D. Kressner, "Numerical methods and software for general and structured eigenvalue problems," Ph.D. dissertation, TU Berlin, Berlin, Germany, 2004.

[33] R. Hecht-Nielsen, "Kolmogorov's mapping neural network existence theorem," in *Proc. Int. Conf. Neural Netw.*, 1987, pp. 11–14.

[34] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102419.

TRUONG THU HUONG (Member, IEEE) received the B.Sc. degree in electronics and telecommunications from the Hanoi University of Science and Technology (HUST), Vietnam, in 2001, the M.Sc. degree in information and communication systems from the Hamburg University of Technology, Germany, in 2004, and the Ph.D. degree in telecommunications from the University of Trento, Italy, in 2007. She came back to work for the Hanoi University of Science and Technology, as a Lecturer in 2009, and became an Associate Professor in 2018. Her research interests include oriented toward network security, artificial intelligence, traffic engineering in next-generation networks, QoE/QoS guarantee for network services, green networking, and development of the Internet of Things ecosystems and applications.

TA PHUONG BAC received the B.Sc. degree in electronics and telecommunications from the Hanoi University of Science and Technology (HUST), Vietnam, in 2020. He has been also a Research Assistant with the Future Internet Laboratory, School of Electronics and Telecommunications, since 2018. His research interests include network security, artificial intelligence, and the Internet of Things ecosystems and applications.

DAO M. LONG is currently a Senior Student of the talented program in electronics and telecommunications engineering with the School of Electronics and Telecommunications, Hanoi University of Science and Technology. He has been a Research Assistant with the Future Internet Laboratory for one year. His research interests include the IoT, network security, machine learning/AI, and its application.

BUI D. THANG is currently a Senior Student of the talented program in electronics and telecommunications engineering with the School of Electronics and Telecommunications, Hanoi University of Science and Technology. He has been working with the Future Internet Laboratory for one year. His research interests include the IoT, network security, machine learning/AI, and its application.

NGUYEN T. BINH is currently a Senior Student of the international program in information and communication technology with the School of Information and Communication Technology, Hanoi University of Science and Technology(HUST). Since 2017, he has been a Research Assistant with the Future Internet Laboratory. His research interests include consist of the Internet of Things, software defined networks, and network architecture.

TRAN D. LUONG is a currently Senior Student of the talented program in electronics and telecommunications engineering with the School of Electronics and Telecommunications, Hanoi University of Science and Technology. He has been a Research Assistant with the Future Internet Laboratory for one year. His research interests include the IoT, network security, machine learning/AI, and its application.

TRAN KIM PHUC received the degree in engineering and the M.Eng. degree in automated manufacturing, and the Ph.D. degree in automation and applied informatics from the Université de Nantes, Nantes, France. He is currently an Associate Professor in automation and industrial informatics with ENSAIT and GEMTEX, Roubaix, France. His research interests include real-time anomaly detection for industrial big data, cybersecurity of industrial systems, and smart healthcare system using the IoT and artificial intelligence. He has published more than 50 articles in peer-reviewed international journals and proceedings of international conferences. He has supervised more than 7 Ph.D. students. In addition, as Project Coordinator, he has conducted two national research projects. He is involved in a European project (FBD-Bmodel - H2020 program) and different regional research projects.