

A Horizontal Federated Learning Approach to IoT Malware Traffic Detection: An Empirical Evaluation with N-BaIoT Dataset

Phuc Hao Do^{***}, Tran Duc Le^{**}, Vladimir Vishnevsky^{****}, Aleksandr Berezkin^{*}, Ruslan Kirichek^{*,****}

^{*}*The Bonch-Bruевич Saint-Petersburg State University of Telecommunications, Saint-Petersburg, Russia*

^{**}*University of Science and Technology – The University of Danang, Da Nang, Viet Nam*

^{***}*Danang Architecture University, Da Nang, Viet Nam*

^{****}*V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, Russia*

haodp@dau.edu.vn, letranduc@dut.udn.vn, vishn@inbox.ru, berezkin.aa@sut.ru, kirichek@sut.ru

Abstract — The increasing prevalence of botnet attacks in IoT networks has led to the development of deep learning techniques for their detection. However, conventional centralized deep learning models pose challenges in simultaneously ensuring user data privacy and detecting botnet attacks. To address this issue, this study evaluates the efficacy of Federated Learning (FL) in detecting IoT malware traffic while preserving user privacy. The study employs N-BaIoT, a dataset of real-world IoT network traffic infected by malware, and compares the effectiveness of FL models using Convolutional Neural Network, Long Short-Term Memory, and Gated Recurrent Unit models with a centralized approach. The results indicate that FL can achieve high performance in detecting abnormal traffic in IoT networks, with the CNN model yielding the best results among the three models evaluated. The study recommends the use of FL for IoT malware traffic detection due to its ability to preserve data privacy.

Keyword — IoT, abnormal traffics, malware detection, federated learning, AI model

I. INTRODUCTION

As a result of the rapid growth of the Internet of Things (IoT) technology, IoT devices have become an integral part of people's daily life. However, it inevitably introduces some network security challenges [1][2]. IoT devices are easy targets for malicious attacks such as malware attacks due to their heterogeneity and vulnerability. The prevalence of privacy and security concerns is rising due to the continual expansion of IoT devices and the resulting exposure of more and more private data online. It is essential to monitor IoT networks to prevent malicious cyberattacks on IoT devices [3]. By studying the traffic of IoT devices, network intrusion detection in the IoT ecosystem may be enhanced, and cyberspace security can be guaranteed [4][5][6].

In recent years, the expansion of deep learning has played a significant role in advancing IoT intrusion detection research [7]. Deep neural networks are employed to automatically identify dataset features, thereby reducing the feature engineering workload and improving data processing performance without requiring human intervention. Despite its benefits, the application of the Internet anomaly detection approach to the IoT is not straightforward due to the large amount of data required. Most existing machine learning or artificial intelligence solutions rely on a single server to collect data from various IoT devices and build global models [8]. However, this approach is not always effective, particularly when device actions involve sensitive or private information that could severely impact environmental security and privacy if disclosed to unauthorized parties in IoT networks.

In the context of preserving information privacy and integrity, Federated Learning (FL) [9] has gained increasing relevance. FL involves decentralizing the training model among several nodes or clients that use local data. Each decentralized node trains a distinct model on its data and distributes the model parameters (not the private data) to others through a central entity known as a server or via a peer-to-peer methodology [10]. The model parameters are then aggregated to produce a singular and global model. After several iterations, each client obtains a global model by

Manuscript received on Jan. 19, 2023. This work was supported by the Russian Science Foundation within of scientific project No. 22-49-02023 and it is also supported by a follow-up of the invited journal the accepted & presented paper entitled "A Horizontal Federated-Learning Model for Detecting Abnormal Traffic Generated by Malware in IoT Networks" of the 25th International Conference on Advanced Communication Technology (ICACT2023)

P. H. Do, he is a Ph.D student in the Department of Communication Networks and Data Transmission at Bonch-Bruевич Saint-Petersburg State University of Telecommunications, Russia. (email: haodp@dau.edu.vn).

T. D. Le, he is a professor in Information Technology Faculty, The University of Danang - University of Science and Technology, Danang, Vietnam (email: letranduc@dut.udn.vn).

V. Vishnevsky, he is a professor at V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Russia. (email: vishn@inbox.ru).

Aleksandr Berezkin, he is working at the Bonch Bruевич Saint Petersburg State University of Telecommunications as the Associate Professor of Department of Programming Engineering and Computer Science. (email: berezkin.aa@sut.ru).

Dr. Sc. Ruslan Kirichek, he is a professor in the Department of Communication Networks and Data Transmission at Bonch-Bruевич Saint-Petersburg State University of Telecommunications, Russia. (corresponding author, phone: +7-921-970-0160, email: kirichek@sut.ru).

aggregating their unique models. This strategy naturally supports data privacy because data is not shared with external identities.

In the paper, we aim to build a horizontal Federated-Learning model to detect abnormal traffics, specifically DDoS attack traffic generated by malware, such as Botnet in IoT networks. To achieve this, we plan to utilize popular models like Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU).

In this study, we utilize the N-BaIoT [11] dataset for model training due to its real-world origin, as opposed to other datasets that simulate network traffic. Our primary objective is to evaluate the efficacy of Federated Learning (FL) in comparison to traditional centralized artificial intelligence models. To achieve this goal, we address research questions such as whether FL can accurately detect malware or malicious traffic in IoT networks and which training models are suitable when applied to this decentralized approach.

Hence, the primary contributions of this study are as follows:

- Employing feature selection and feature engineering techniques to preprocess the N-BaIoT dataset and select features that have the most considerable impact on the model's accuracy and performance;
- Deploying and evaluating various deep learning models, including CNN, LSTM, and GRU, in the FL approach with different dataset portions to determine their compatibility and select the most suitable model for the FL approach;
- Conducting a comparative analysis between the FL model and traditional deep learning models using the same dataset, thus assessing the feasibility of FL in detecting abnormal traffic in IoT networks.

The results of this study will be included in the Draft Recommendation ITU-T Q.TSRT_IoT "Test specifications for remote testing of Internet of Things using the probes".

II. RELATED WORKS

Several researchers have conducted a thorough investigation to find a more trustworthy anomaly detection strategy for IoT networks. To solve the issue of constrained device resources, the study in [12] created a unique, lightweight attack detection method that uses the Support Vector Machines (SVM) algorithm to recognize hazardous assaults in the IoT space. To offer a distributed intrusion detection system, Ferdowsi et al. recommended anomaly detection utilizing distributed Generative Adversarial Networks (GANs) [13]. Each device may protect user privacy while applying a detection model to its data since it is not maintained centrally. A centralized IoT intrusion detection system based on fog computing was suggested in the publication [14]. Two cascading recurrent neural networks (RNNs) are used in the design, each with its hyperparameters and attack-specific tuning. The system administrator is notified if any of the two RNNs determines that an input instance is malicious. Time-series data from IIoT sensors were utilized by the authors of [15] to identify abnormalities using an attention-based convolutional neural network with

LSTM [16].

In recent years, FL has risen to prominence in cybersecurity. This IoT security paradigm has already been used in several works. In this context, the study proposal [17] underlined the problem with traditional AI-based solutions' lack of data privacy. However, a private dataset was used for the examination. Despite sharing similar objectives, the research mentioned in [18, 19] mainly focused on industrial IoT devices. They looked at application samples and sensor readings rather than network data.

The authors introduced the Federated Averaging (FedAVG) approach in [20], which is a strong basis for many FL-based investigations. In this strategy, a central server coordinates the training of a global model across several clients using their datasets. The server is responsible for averaging the client-sent models' parameters and sending the resulting global model back to the clients. Until a termination condition is met, this process is repeated. FL has significantly advanced since several researchers [21] examined the most current advancements in this area.

Two more robust model aggregation functions were suggested in [22]. They are based explicitly on the coordinate-wise mean and median of the clients' models supplied to the server. The authors of [23] suggested resampling as a way to lessen the variability in the distribution of the models that the clients supplied. It should be used before employing a powerful aggregating function. When used with models trained on non-IID datasets, it seeks to lessen any negative consequences that such a function could have.

The paper [24] focuses on the challenging task of optimizing neural architectures in the federated learning framework. The authors describe recent work on federated neural architecture search, which involves searching for optimal neural architectures across multiple clients in a distributed manner. They categorize these approaches into online and offline implementations, as well as single- and multi-objective search approaches. They also explain the different types of federated learning, including horizontal, vertical, and hybrid.

The authors [25] propose a Federated Learning (FL) based IoT Traffic Classifier (FLITC) that uses Multi-Layer Perception (MLP) neural networks to classify traffic data while keeping local data on IoT devices. FLITC sends only the learned parameters to the aggregation server, reducing communication costs and latency. The main idea behind FLITC is to train a shared model on distributed devices without exchanging raw data, thereby preserving privacy and reducing communication overhead. The method is expected to improve the accuracy and efficiency of IoT traffic classification while maintaining data privacy.

The study [26] proposes a federated-learning traffic classification protocol (FLIC) for Internet traffic classification without compromising user privacy. FLIC aims to achieve accuracy comparable to centralized deep learning for Internet application identification without privacy leakage. It can classify new applications on-the-fly when a participant joins the learning process with a new application, which has not been done in previous works. The authors implemented the FLIC prototype using TensorFlow, allowing

clients to gather packets, perform on-device training, and exchange training results with the FLIC server. They demonstrated that federated learning-based packet classification achieves an accuracy of 88% under non-independent and identically distributed (non-IID) traffic across clients. When a new application was added dynamically as a client participating in the learning process, an accuracy of 92% was achieved.

The study [27] focuses on the security challenges posed by the deployment of IoT devices and proposes a federated learning-based edge device identification (FedeEDI) method to control access and manage internal devices. The authors note that external attackers often exploit vulnerable IoT devices to gain access to the target's internal network and cause security threats. They review existing literature on deep learning-based algorithms for edge device identification and highlight the limitations of centralized learning-based EDI (CentEDI) methods that train all data together. The authors propose a federated learning-based approach that addresses data security concerns and is suitable for deployment on edge devices.

Overall, the studies presented in this section highlight the potential of federated learning as a viable solution to the challenges posed by the deployment of IoT devices.

III. THE HORIZONTAL FEDERATED LEARNING

A. Horizontal FL

Federated learning can be divided into three categories: horizontal federated learning, vertical federated learning, and federated transfer learning [28].

Horizontal federated learning is designed for scenarios where participating clients' datasets share the same feature space but contain different samples. The term "horizontal" originates from instance-distributed learning, as illustrated in Fig. 1a, where datasets are horizontally partitioned across data samples and allocated to clients. In federated learning, data can be considered horizontally partitioned when different clients generate data with the same attributes (features) but distinct samples. Similarly, as indicated by the part surrounded by the two dashed lines in Fig. 1b, the data can be considered horizontally partitioned in federated learning when different data are generated on different clients that have the same attributes (features). For example, two hospitals in different regions may have distinct patients but perform the same tests and collect the same personal information (e.g., name, age, gender, and address). There are three main differences between instance-distributed learning and horizontal federated learning [29]:

- Data are typically independent and identically distributed (IID) in distributed learning but may be non-IID in horizontal federated learning. Designers can manually allocate subsets of client data to be IID in distributed learning to enhance convergence, while in horizontal federated learning, the central server has no access to raw data, which is usually non-IID on different clients link.springer.com.
- Horizontal federated learning involves a large number of connected clients, whereas instance-distributed learning

often does not have as many workers. Too many workers may worsen distributed training performance when the total data amount is fixed.

- Global model update mechanisms differ. In instance-distributed learning, a deep neural network synchronously updates the global model once local gradients of mini-batch data are calculated. This approach is not suitable for horizontal federated learning due to communication constraints.

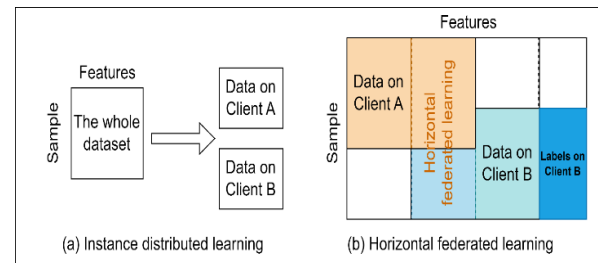


Fig. 1. (a) Instance distributed learning
(b) horizontal federated learning

Horizontal federated learning faces three main challenges compared to centralized learning: reducing communication resources, improving convergence speed, and ensuring no private information leakage.

In contrast, vertical federated learning is applicable when datasets share the same sample space but have different feature spaces. Vertical federated learning is similar to feature distributed learning, where the central server acts as a coordinator to compute the total loss instead of aggregating uploaded weights.

Federated transfer learning, on the other hand, leverages vertical federated learning with a pre-trained model trained on a similar dataset for solving a different problem.

Typical horizontal federated learning (Fig. 2) algorithms, such as the FedAvg, consist of the following main steps:

- Initialize the global model parameters on the server and download the global model to every participating (connected) client.
- Every connected client learns the downloaded global model on its own data for several training epochs. Once completed, the updated model parameters or gradients (gradients here mean the difference between the downloaded model and the updated model) would be sent to the server. Note that the clients may have different amounts of training data and unbalanced computational resources. As a result, the server is not able to receive the uploads from different clients at the same time.
- The server aggregates the received uploads (synchronously or asynchronously) to update the global model.
- Repeat the above two steps until convergence.

From the above steps, we can find that the central server can only receive model weights or gradients of the participating clients and has no access to any local raw data. Therefore, users' privacy is immensely protected in horizontal federated learning.

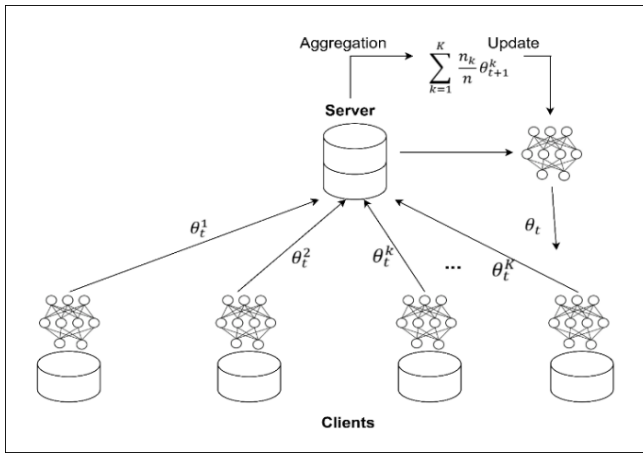


Fig. 2. Flowchart of federated learning. θ is the global model parameters, n_k is the data size of client k , K is the total number of clients and t is the communication round in federated learning. We initialize global model parameters randomly at the beginning of the communication round and use updated model parameters afterward

In this study, the use of this model is suitable the fact that we use a single dataset, which is N-BaIoT. Thus the features are the same.

The Fig 3 displays the approach used in this study, which consists of the dataset, data processing, data aggregation, divide “attribute class”, and classifier. Standardization and minimum-maximum normalization min-max normalization) are also used as data preparation techniques.

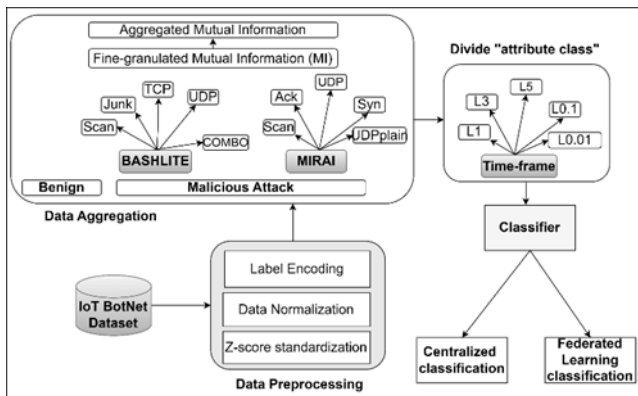


Fig. 3. The research flow

B. Dataset

There are many public datasets related to IoT security as follows:

- N-BaIoT¹: This dataset tackles the shortage of available botnet datasets, particularly for the Internet of Things. It implies actual traffic data, collected from nine commercial IoT devices that were actually infected with Bashlite and Mirai malware.
- MedBIoT²: It is collected from actual and simulated Internet of Things devices in a medium-sized network (i.e., 83 devices). The data collection is divided into categories based on the kind of traffic (malware or normal traffic), making it simple to label the data and extract characteristics from the raw pcap files.

- Bot-IoT³: The BoT-IoT dataset was developed using a realistic network environment constructed at the Cyber Range Lab of the UNSW Canberra Cyber Center. The dataset consists of Service Scan, DDoS, DoS, Keylogging, and Data exfiltration attacks. In addition, DDoS and DoS attacks are categorized based on the protocol employed.
- TON-IoT⁴: This dataset contains heterogeneous data sources gathered from Telemetry datasets of IoT and IIoT sensors, Windows OS, Ubuntu, and network traffic datasets. The datasets were acquired from the Cyber Range and IoT Labs' realistic and wide network.
- IoT-23⁵: IoT-23 contains 20 malware captures (scenarios) conducted on IoT devices, and 3 benign IoT traffic captures. The authors ran a particular malware sample on a Raspberry Pi for each malicious scenario, which utilized multiple protocols and carried out distinct tasks.

The N-BaIoT dataset is collected separately for each device, so it is very suitable to implement the FL empirical model. Therefore, we will use it for our research. IoT devices in the N-BaIoT dataset were targeted by the Bashlite and Mirai botnet attack families. Each file has 115 features as well as a class label. The dataset has also been built to support binary and multi-class classification. The target class labels are: "TCP attack," "benign" for detection, and "Mirai" or "Bashlite" attack types for multi-class classification.

The N-BaIoT dataset comprises feature headers that describe various aspects of network traffic data, which can be grouped into categories based on their purpose and scope. The headers related to stream aggregation include H, which provides statistics summarizing recent traffic from the packet's host (IP); HH, which summarizes recent traffic from the packet's host to the packet's destination host; HpHp, which summarizes recent traffic from the packet's host+port (IP) to the packet's destination host+port; and HH_jit, which summarizes the jitter of the traffic from the packet's host to the packet's destination host. Additionally, the time-frame decay factor Lambda determines how much recent history of the stream is captured in these statistics, with values such as L5, L3, L1, and others.

The dataset also provides statistics extracted from the packet stream, including the weight of the stream, which can be viewed as the number of items observed in recent history. The mean represents the average value of the stream, while the standard deviation is denoted by std. The root squared sum of the two streams' variances is represented by radius, while the root squared sum of the two streams' means is denoted by magnitude. Moreover, an approximated covariance between two streams is represented by cov, and an approximated covariance between two streams is denoted by pcc.

Overall, the N-BaIoT dataset provides a comprehensive set of feature headers that can be used to analyze various aspects of network traffic data, which can be useful for developing and evaluating intrusion detection systems in IoT networks.

In this study, we will focus on multi-classification. We use part of the dataset to proceed with the experiment. The multi-class classification will be evaluated.

¹ <https://www.kaggle.com/datasets/mkashifn/nbaiot-dataset>

² <https://cs.taltech.ee/research/data/medbiot/>

³ <https://iee-dataport.org/documents/bot-iot-dataset>

⁴ <https://research.unsw.edu.au/projects/toniot-datasets>

⁵ <https://www.stratosphereips.org/datasets-iot23>

C. Preprocessing

Although data preparation [30] is difficult and time-consuming [31], its importance has been demonstrated for speeding the training process and enhancing its effectiveness. Consequently, this study employs label encoding, min-max normalization, and standardization as pre-preprocessing procedures.

1) Label Encoding

The class label has 11 different category values (one "Benign" class and ten subclasses of attack type). As a result, before these attributes are used with the models, they are converted into numerical values. There are several methods for converting categorical values, including one-hot encoding [32], ordinal encoding [33], similarity encoding [34], entity embedding [35], and multi-hot encoding. Among them, one-hot encoding and ordinal encoding are the most widely employed. This study uses the one-hot encoding technique for encoding categorical values.

2) Normalization and Standardization

Suppose the columns in a dataset contain values with varying ranges. In that case, the performance of both regression and classification models is negatively impacted. Mahfouz et al. in [36] demonstrated how this issue causes the performance of the models to decrease when uneven scales of features are observed in a dataset. It is required to determine the acceptable range for the insignificant and dominating values to address such problems. The two most used methods are min-max normalizing and z-score standardization:

- The following equation is used to apply min-max normalization to change the values of the dataset's feature values into the range [0, 1]:

$$X_{normalized} = \frac{X - X_{min_value}}{X_{max_value} - X_{min_value}} \quad (1)$$

where $X_{normalized}$ represents the normalized value, X_{min_value} and X_{max_value} are the intended interval's boundary range, which is [0, 1], and X is the initial value that would be altered inside those ranges.

- Z-score standardization is used to rescale dataset features, reflecting the characteristics of a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$.

$$X_{normalized} = \frac{X - \mu}{\sigma} \quad (2)$$

3) Data aggregation

After encoding the target class, the "benign" class was added to the N-BaIoT dataset, which has 115 characteristics and 10 class labels. To improve the performance of the classification process, we can implement some feature selection methods. In the scope of the paper, we choose the Mutual Information (MI) method for the feature selection process of the input data.

An aggregated MI with multiple rank aggregation functions is developed and evaluated for the multi-class

dataset. The concept of aggregated MI is explained as follows:

- Calculate the information gain score for each feature, f_i , in dataset D relative to class type $c \in C$. The features are then ranked based on the aggregator functions listed below. Calculate the information gain score for each feature, f_i , in dataset D relative to class type $c \in C$. After that, the features are ranked based on the aggregation methods (Min, Max, Mean). Only part of preserved features are supplied to the classifiers, and the total performance is assessed.
- List of aggregators:
 - o *Min*: Chooses the class type c_i as the target class and takes the relevance score with the lowest value.
 - o *Max*: Chooses the class type c_i as the target class and takes the relevance score with the highest value.
 - o *Mean*: Chooses the class type c_i as the target class and takes the relevance score with the mean value.

4) Dividing attribute class

In this research study, we aim to evaluate the classification performance of the N-BaIoT dataset by dividing its attributes into different classes based on the time-frame property. The dataset includes time-frames such as L5, L3, L1, L0.1, and L0.01, which correspond to the traffic capture time. We will utilize these time frames to create attribute sets for each subclass. Specifically, Table I shows the attribute set for the time frame L5, which contains 23 attributes. We will follow the same procedure for the remaining time frames. This approach allows us to assess the classification performance of each class containing different attributes, providing insights into the effectiveness of the dataset's features for anomaly detection in IoT networks.

TABLE I
SOME PROPERTIES OF THE TIME-FRAME (L5)

No	Attribute	No	Attribute
1	MI_dir_L5_weight	13	HH_L5_pcc
2	MI_dir_L5_mean	14	HH_jit_L5_weight
3	MI_dir_L5_variance	15	HH_jit_L5_mean
4	H_L5_weight	16	HH_jit_L5_variance
5	H_L5_mean	17	HpHp_L5_weight
6	H_L5_variance	18	HpHp_L5_mean
7	HH_L5_weight	19	HpHp_L5_std
8	HH_L5_mean	20	HpHp_L5_magnitude
9	HH_L5_std	21	HpHp_L5_radius
10	HH_L5_magnitude	22	HpHp_L5_covariance
11	HH_L5_radius	23	HpHp_L5_pcc
12	HH_L5_covariance		

D. Deep Learning Techniques Applied to Abnormal Traffic Detection

1) Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) [37]: CNNs were initially developed for image recognition tasks, but researchers have adapted them to process textual data in phishing detection with great success. CNNs utilize convolutional layers to automatically learn features and patterns from the email content by applying multiple filters to different regions of the input text. These filters can capture local patterns, such as word groupings or specific textual structures, which can indicate phishing attempts.

The formula for CNN classification varies depending on the specific architecture and design of the CNN model. However, a general formula for CNN classification can be broken down into the following steps:

- Convolutional Layers: The data input is passed through one or more convolutional layers. Each convolutional layer applies a set of filters to the data input, creating feature maps that capture different aspects of the data.
- Activation Function: After each convolutional layer, an activation function is applied to introduce non-linearity to the output feature maps. The most commonly used activation function is ReLU (Rectified Linear Unit).
- Pooling Layers: After the activation function, the feature maps are passed through one or more pooling layers. Pooling layers downsample the feature maps by taking the maximum or average value within a specific window size. This reduces the size of the feature maps and makes the model more computationally efficient.
- Flatten: After the final pooling layer, the feature maps are flattened into a one-dimensional vector.
- Fully Connected Layers: The flattened feature vector is then passed through one or more fully connected layers. Each fully connected layer applies a set of weights to the input vector and outputs a new vector of a specified size.
- Softmax: The final fully connected layer uses the softmax function to convert the output vector into a probability distribution over the different classes in the classification task. The class with the highest probability is then selected as the predicted class.

The above steps can be represented as a mathematical formula for a basic CNN classification model as follows:

$$y = sm(W_2 * relu(W_1 * pool(conv(x)) + b_1) + b_2) \quad (3)$$

where x is the data input, $conv$ represents the convolutional layers, the $pool$ represents the pooling layers, W_1 and b_1 represent the weights and biases of the first fully connected layer, $relu$ represents the activation function, W_2 and b_2 represent the weights and biases of the final fully connected layer, and sm is softmax function, softmax function represents the final activation function that outputs the probability distribution over the different classes.

2) Long Short-Term Memory (LSTM)

LSTM [38] is a recurrent neural network (RNN) that uses feedback to remember portions of the input and make predictions. RNNs are intended to process sequential input and have thus found widespread use in speech recognition and machine translation. The well-known problem of vanishing gradients affects the long-term memory of conventional RNNs. It restricts their ability to make predictions based on the most recent data in the sequence. LSTM (Fig. 4) overcomes the problem of vanishing gradients and can thus handle longer sequences (long-term memory). LSTM can extract context from a succession of features. Using a gate function, it can add or delete information from the hidden state vector, preserving vital information in the hidden layer vectors.

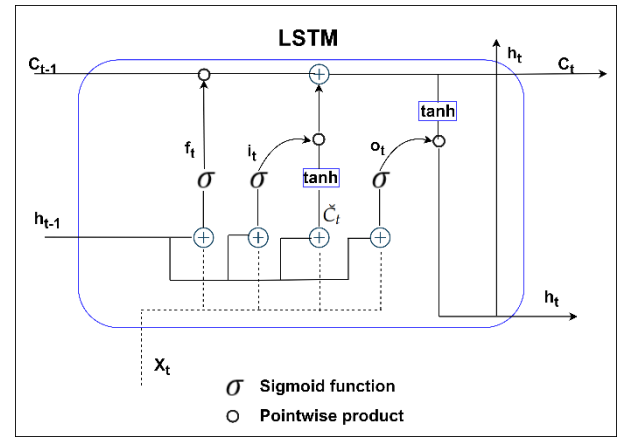


Fig. 4. LSTM Model

There are three gate functions in LSTM: the forget, the input, and the output. The forget gate is used to regulate how much information from C_{t-1} is maintained throughout the computation of C_t , and i_t the forget vector may be written as follows:

$$f_t = \sigma(U^f x_t + W^f h_{t-1} + b_f) \quad (4)$$

where U^f , W^f , and b_f are the forget gate's parameters, x_t is the input vector in step t , and h_{t-1} is the hidden state vector in the previous step. The input gate determines the amount of x_t information added to C_t and may be stated as follows:

$$i_t = \sigma(U^i x_t + W^i h_{t-1} + b_i) \quad (5)$$

where U^i , W^i , and b_i are the input gate's parameters and C_t may be determined by using both the input gate's vector i_t and the forget gate's vector f_t as shown below:

$$\tilde{C}_t = \sigma(U^c x_t + W^c h_{t-1} + b_c) \quad (6)$$

Where $\tilde{C}_t = \tanh(U^c x_t + W^c h_{t-1} + b_c)$ reflected the information represented by the vector of the hidden layer. Note that $*$ represents the Hadamard product (element-wise). The output gate regulates the output in C_t , and:

$$o_t = \sigma(U^o x_t + W^o h_{t-1} + b_o), h_t = o_t * \tanh(C_t) \quad (7)$$

where U^o , W^o , and b_o are the output gate parameters and C_t is the internal state at time step t .

3) Gated Recurrent Unit (GRU)

A GRU model [39] (Fig. 5) is also an RNN and LSTM variant. However, GRU has only two gates: the update and the reset gates. Due to its simplicity and ease of training, GRU is superior to LSTM, in which data transmitted to the output is decided by the gates, which are two vectors. The update gate helps the model to calculate how much information from the past must be transmitted to the future. For step t , the update gate z_t is computed using the following formula:

$$z_t = \sigma(U^z x_t + W^z h_{t-1}) \quad (8)$$

where U^z , W^z are the weights of the update gate and h_{t-1} stores data for the previous $(t-1)$ units. The reset gate is used to determine how much of the past data to forget, which is computed using the following formulas:

$$r_t = \sigma(U^r x_t + W^r h_{t-1}) \quad (9)$$

where U^r , W^r are the reset gate's weights and h_{t-1} holds information for the previous $(t-1)$ units. Using the reset gate, the current memory content will preserve the essential information from the past:

$$c_t = \tanh(U^c x_t + r_t * W^c h_{t-1}) \quad (10)$$

where U^c , W^c are the weights. Note that $*$ represents the Hadamard product (elementwise). The final step involves calculating the vector h_t that contains the information for the current unit and transmits it along the network:

$$h_t = z_t * h_{t-1} + (1 - z_t) * c_t \quad (11)$$

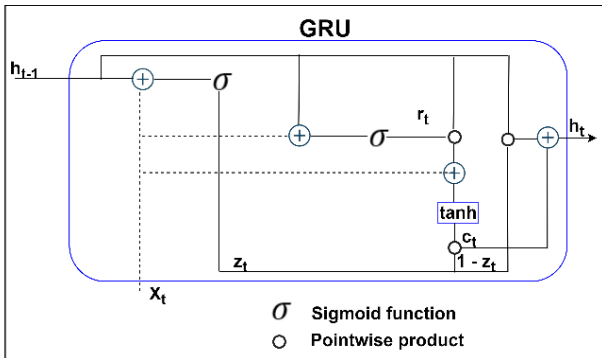


Fig. 5. GRU Model

A GRU network acquires a long spatial (or temporal) sequence with less computational complexity than a conventional encoder-decoder. GRU can handle longer sequences than regular RNNs because it can solve the vanishing gradient problem with gating techniques. To build context between features spread out across a large area or assess the degree of interconnections between features, GRU and LSTM may be used for sequences of spatial data.

4) Experiment Flow for Centralized Approach

We should recall that we use the N-BaIoT dataset, in which abnormal traffic is caused by malware, especially Botnets. Therefore, here we use "Botnet" as a general concept to indicate the source of abnormal traffic.

The Fig 6 shows the operational flow of the experimental process of Botnet detection in the centralized approach. Data is centralized in one place. First, we perform data processing and feature selection

from the dataset data to process the data and select the attributes that are good enough to include in the training model. The dataset is divided into 70% for training, 10% for validation, and 20% for testing.

After that, we will apply three models, CNN, LSTM, and GRU, to detect abnormal traffic generated by Botnet.

The results are obtained, evaluated, and compared between these models.

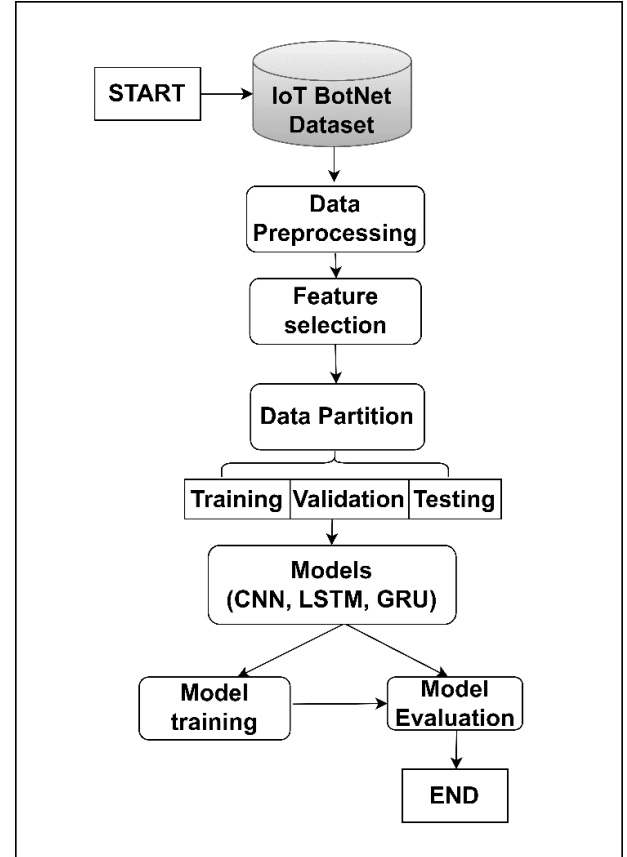


Fig. 6. Experiment Flow for Centralized Approach

E. Federated learning Model

This section describes the architecture of the FL approach. We will present the components and how they interact during the training and testing of models. In addition, it illustrates how the framework is implemented for our validation use case, which uses the N-BaIoT dataset. The Fig 7 depicts the framework architecture, which consists of K clients, each of which owns data from a single device and a server that coordinates the FL process. And the Fig 8 shows the back-and-forth interaction between the client-server and the preprocessing and feature selection process.

1) Model training component

Two FL techniques derived from the well-known FedAVG, Mini-batch aggregation, and Multi-epoch aggregation, are considered. The primary distinction between FedAVG and the other algorithms is that FedAVG considers the aggregation function as a parameter. As a result, the server can experiment with aggregating methods other than average. In this paper, we will implement the multi-epoch aggregation method. The model is trained for all E epochs simultaneously before

being transmitted to the server for multi-epoch aggregation.

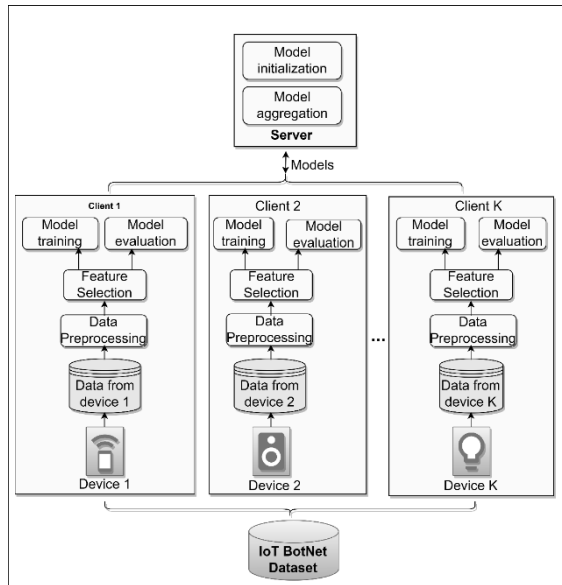


Fig. 7. FL framework architecture and its components

According to [20], averaging models may produce arbitrarily poor results due to the objective's non-convexity. This issue is significantly more likely to arise with multi-epoch aggregation since the models are trained individually for a much more extended time before being combined. To minimize this issue, multi-epoch aggregation training is performed for $T = 30$ rounds with a decreasing learning rate.

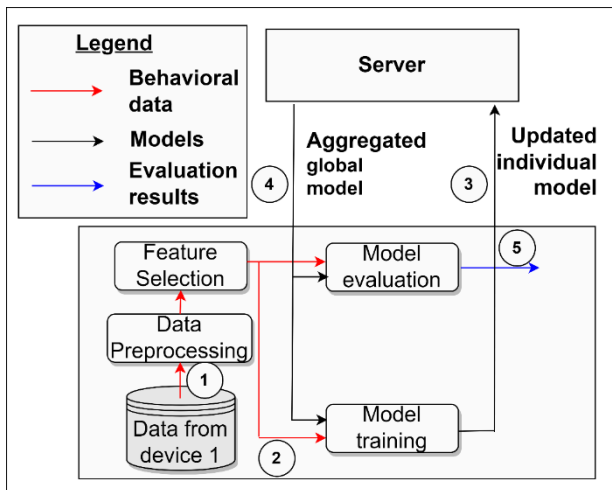


Fig. 8. Detailed view of the training process in the FL

2) Server component

The server is responsible for organizing the clients' training activities. In particular, it initializes the model from scratch. It compiles the models that the clients send into a so-called global model. The server is responsible for initializing the initial model's weights. All clients will have access to the first model when it is finished, and training may begin. It is important to note that every client starts with the same model. The server must combine the updated parameters from each client after receiving them to create the new global model parameters.

F. Evaluations

Using the N-BaIoT dataset, this experiment aims to evaluate how well our approach discovers abnormal IoT traffic caused by malware. It is crucial to compare the federated learning technique with conventional solutions to ensure that it is appropriate for IoT networks.

This research study evaluates the effectiveness of three well-known models, CNN, LSTM, and GRU, using three different approaches for anomaly detection in IoT networks. The first approach is a centralized approach that uses all attributes of the dataset. The second approach is Federated Learning with all attributes, while the third approach is Federated Learning with the division of attributes based on the time-frame characteristics of the dataset. The time-frame characteristics divide the attributes into five groups, corresponding to L1, L3, L5, L0.1, and L0.01, based on the time intervals of the collected streams. The evaluation provides insights into the performance of the different models and training approaches for anomaly detection in IoT networks, which can inform the development of more accurate and efficient intrusion detection systems.

The Fig 9 shows the accuracy of the centralized model based on all attributes of the dataset. The CNN model achieves the highest accuracy of 90.9%, while the LSTM and GRU models reach their highest accuracies of 88.39% and 90.66%, respectively. However, it is important to note that the LSTM and GRU models have more variability in their accuracies across epochs than the CNN model. For example, the LSTM model's accuracy fluctuates between 46.08% and 88.39%, while the GRU model's accuracy ranges from 56.5% to 90.9%. Furthermore, the accuracy of the LSTM model drops significantly after epoch 14, whereas the CNN and GRU models continue to perform well. The GRU model also shows a significant improvement in accuracy from epoch 1 to epoch 3, indicating that it may require more epochs to converge. Overall, the results suggest that the CNN model is the most reliable and consistent performer, while the LSTM and GRU models may require further optimization and fine-tuning to achieve optimal accuracy.

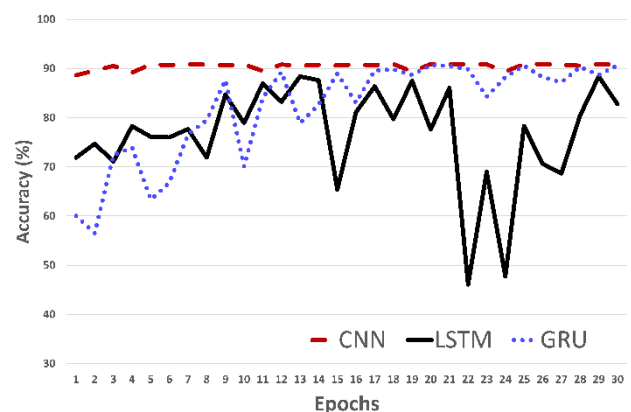


Fig. 9. The performance of the centralized learning model based on all attributes of the dataset

The Fig 10 shows the loss value when training the model with centralized data corresponding to CNN, GRU, and LSTM algorithms. It is interesting to note that the loss values of the LSTM and GRU models are much higher than those of the CNN model, especially in the earlier epochs. This could be due to the fact that the LSTM and GRU models are more complex than the CNN model, and therefore require more

training epochs to converge to a similar level of accuracy. It is also worth mentioning that the loss values of the LSTM model show significant fluctuations throughout the training process, with some epochs having much higher losses than others. This could be an indication that the LSTM model is more sensitive to the specific data samples used for each epoch of training. Overall, the CNN model achieves the lowest loss values consistently across all epochs, followed by the LSTM model and then the GRU model. This suggests that the CNN model is the most effective at reducing the error between predicted and actual values during training.

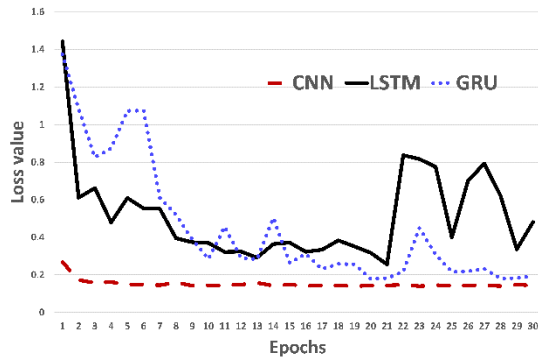


Fig. 10. Loss value of centralized learning model based on all attributes of the dataset

The Federated Learning approach's results of model training based on all attributes of the dataset with distributed data on each client are shown in Fig 11.

In Fig. 11, with the FL approach, the CNN model performs the best with the highest accuracy scores in most of the epochs. It starts with an accuracy of 85.017% in the first epoch and reaches up to 90.831% in the 25th epoch before slightly decreasing in the last few epochs.

On the other hand, the LSTM model has significantly lower accuracy scores compared to the CNN model in most of the epochs. It starts with an accuracy of 33.997% in the first epoch and gradually improves before reaching its peak at 88.803% in the 29th epoch. However, it shows a significant fluctuation in its performance throughout the epochs, with some epochs having a sharp drop in accuracy.

Similarly, the GRU model also has lower accuracy scores compared to the CNN model, but it performs better than the LSTM model in most of the epochs. It starts with an accuracy of 35.42% in the first epoch and gradually improves before reaching its peak at 89.56% in the last epoch. However, like the LSTM model, it also shows significant fluctuation in its performance in some epochs. Overall, the CNN model seems to perform better than the LSTM and GRU models in terms of classification accuracy on this dataset.

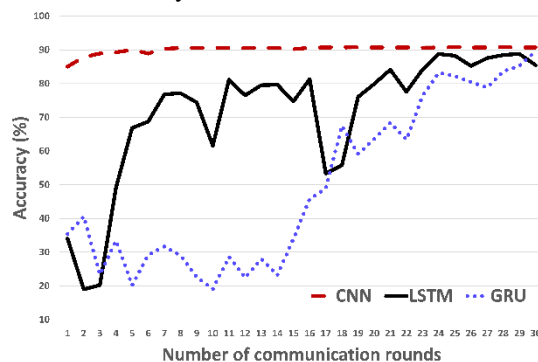


Fig. 11. Classification performance of the FL model based on all attributes of the dataset

The Fig 12 shows the loss value when training distributed data with the FL approach using all attributes. We can see that the loss values of CNN decrease gradually with each epoch, indicating that the model is improving in performance. On the other hand, the loss values for LSTM and GRU are more erratic, with occasional spikes in loss value. This could be due to the more complex nature of these models.

It is also worth noting that the loss values for LSTM and GRU are generally higher than those of CNN, suggesting that CNN is better suited for this particular task. Overall, the results suggest that CNN outperforms LSTM and GRU in terms of loss value.

Observing the accuracy of the models when deployed for centralized and distributed data, we see that the accuracy does not differ too much. The value fluctuates around 1% for accuracy. However, with centralized data, data privacy cannot be guaranteed. So in the case of ensuring data privacy, especially in IoT networks, the Federated Learning model will respond better.

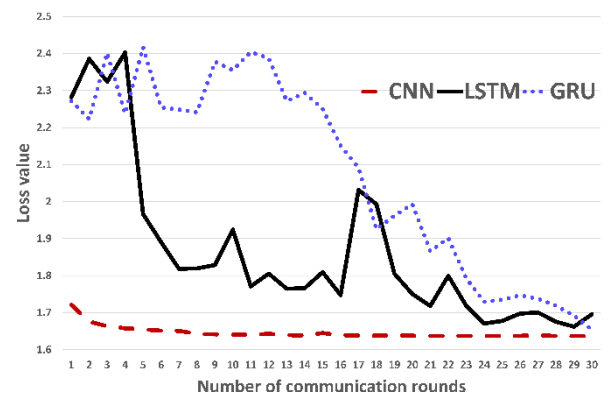


Fig. 12. Loss value of FL model based on all attributes of the dataset

The Fig 13 compares the training times of the centralized and FL-based approaches. The results show that the training time of the model when the data is concentrated is much better. Obviously, the distributed data training adds transmission, computation, and configuration update time for all clients so that the execution time will be longer.

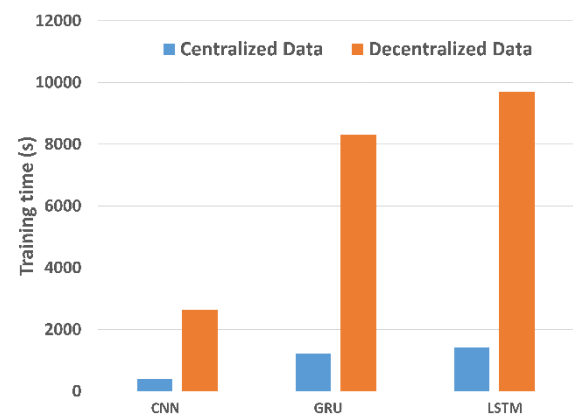


Fig. 13. Time training between centralized and decentralized based on all attributes of the dataset

Table II presents a comparison of three different algorithms, CNN, LSTM, and GRU, based on the metrics of precision, recall, F1-score, and accuracy, for centralized learning based on a time frame of L0.01.

According to the table, the CNN algorithm outperforms the LSTM and GRU algorithms in all four metrics, achieving the highest precision (0.94), recall (0.90), F1-score (0.87), and accuracy (0.90) compared to LSTM and GRU. LSTM achieves the second-best performance with a precision of 0.84, recall of 0.89, F1-score of 0.85, and accuracy of 0.89. GRU, on the other hand, performs the worst among the three algorithms, achieving a precision of 0.78, recall of 0.76, F1-score of 0.73, and accuracy of 0.76. The results suggest that the CNN algorithm is the most suitable for centralized learning based on a time frame of L0.01, as it achieves the highest accuracy and the best balance between precision and recall.

TABLE II
COMPARASION TABLE OF THE ALGORITHMS FOR CENTRALIZED
LEARNING BASED ON TIME FRAME (L0.01)

Metrics	CNN	LSTM	GRU
Precision	0.94	0.84	0.78
Recall	0.90	0.89	0.76
F1-score	0.87	0.85	0.73
Accuracy	0.90	0.89	0.76

In Fig 14, the classification results using the FL model and based on the set of attributes of the time-frame L0.01 of the dataset, the results show that the CNN model achieves the highest accuracy among the three models with an average of 89.44% across all 30 epochs. LSTM and GRU models achieve significantly lower accuracy compared to CNN, with LSTM averaging 75.89% and GRU averaging 76.92% accuracy across all epochs.

It is also observed that the accuracy of all three models tends to increase over the course of the 30 epochs, with fluctuations in some epochs. However, the overall trend shows an increasing trend, with CNN having the most consistent increase in accuracy over time.

Additionally, it is worth noting that the accuracy of the LSTM and GRU models experiences more fluctuations compared to CNN. This may suggest that CNN is more stable and reliable than LSTM and GRU in this particular classification task.

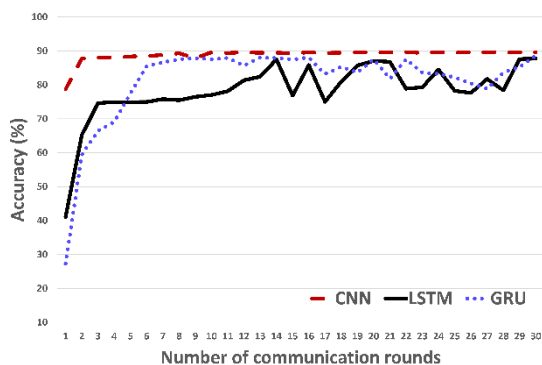


Fig. 14. Classification performance of the FL model based on class attribute L0.01 of the dataset

Table III shows the Precision, Recall, F1-score, and accuracy measures of CNN, LSTM, and GRU algorithms when training centralized data using only L0.1 time-frame algorithms. i.e. use a time frame of 0.1(s) to collect the stream information. The results show that the CNN algorithm gives the best results with an accuracy of 90%, while the LSTM algorithm gives relatively low results, only about 78%. As for

the GRU algorithm, the classification result is 88%. Based on the experimental results, using only the attributes of the time frame L0.1 gives quite similar results to using all the attributes of the data set. However, with the L0.1 time-frame attribute set, the LSTM algorithm gives quite low results, while the CNN and GRU algorithms give many good results.

TABLE III
COMPARASION TABLE OF THE ALGORITHMS FOR CENTRALIZED
LEARNING BASED ON TIME FRAME (L0.1)

Metrics	CNN	LSTM	GRU
Precision	0.93	0.69	0.89
Recall	0.90	0.78	0.88
F1-score	0.87	0.70	0.85
Accuracy	0.90	0.78	0.88

In Fig 15, the classification results using the FL model and based on the attribute set of the time-frame L0.1 of the data set, the results show that the CNN algorithm gives the best results and reaches the value close to 90 % and converge very quickly starting from round 2. As for the LSTM algorithm, the results fluctuate quite a lot through each round, the learning ability of the LSTM model for this L0.1 time-frame attribute set Not good. Experiments show that using the attribute set of this time-frame L0.1 also gives quite similar results to using all the attributes of the data set.

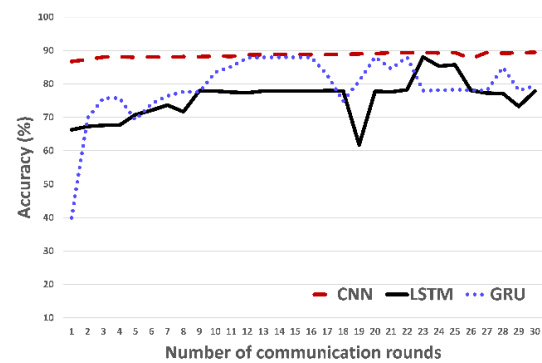


Fig. 15. Classification performance of the FL model based on class attribute L0.1 of the dataset

Table IV presents the evaluation metrics of Precision, Recall, F1-score, and accuracy for the CNN, LSTM, and GRU algorithms when trained on centralized data using L1 time-frame algorithms. L1 refers to a time frame of 1 second to gather the stream's information. According to the findings, the CNN algorithm performed the most accurately, with an accuracy rate of 89%, while the LSTM and GRU algorithms also produced satisfactory results, with accuracy rates of 88% and 86%, respectively. Moreover, these outcomes indicate that utilizing only the L1 time-frame attributes can yield comparable results to those obtained by using the complete set of dataset attributes.

TABLE IV
COMPARASION TABLE OF THE ALGORITHMS FOR CENTRALIZED
LEARNING BASED ON TIME FRAME (L1)

Metrics	CNN	LSTM	GRU
Precision	0.91	0.83	0.82
Recall	0.89	0.88	0.86
F1-score	0.85	0.85	0.82
Accuracy	0.89	0.88	0.86

In Fig 16, the performance of the Federated Learning (FL) model is evaluated based on the L1 time-frame attributes of the dataset. The results indicate that the CNN algorithm outperforms the LSTM and GRU algorithms, achieving an accuracy of 90% with a fast convergence rate, starting from round 4. In contrast, the GRU algorithm's performance fluctuates considerably throughout each round, demonstrating poor learning ability for the L1 time-frame attribute set. Furthermore, the study suggests that using this L1 time-frame attribute set can yield results similar to those obtained when using all the attributes of the dataset.

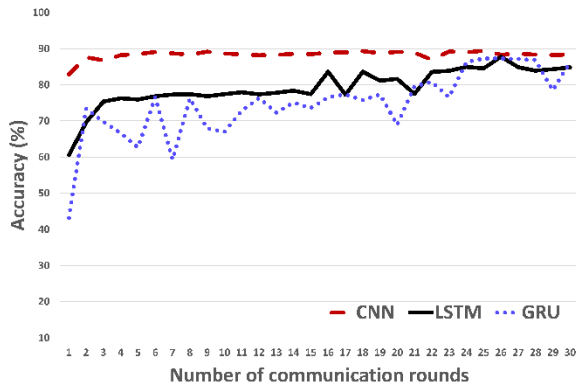


Fig. 16. Classification performance of the FL model based on class attribute L1 of the dataset

Table V presents the evaluation metrics of the CNN, LSTM, and GRU algorithms when trained on centralized data using only the L3 time-frame attribute set, which collects information from a time frame of 3 seconds.

The experimental results indicate that the CNN and LSTM algorithms achieve the highest accuracy of 88%, while the GRU algorithm yields comparatively low results of around 78% when trained on the L3 time-frame attribute set. These findings suggest that using the L3 time-frame attribute set can yield similar results to those achieved with the entire dataset.

TABLE V
COMPARASION TABLE OF THE ALGORITHMS FOR CENTRALIZED
LEARNING BASED ON TIME FRAME (L3)

Metrics	CNN	LSTM	GRU
Precision	0.90	0.83	0.73
Recall	0.88	0.88	0.78
F1-score	0.85	0.84	0.70
Accuracy	0.88	0.88	0.78

In Fig 17 depicts the classification results obtained using the FL model and the L3 time-frame attribute set of the data set. The results indicate that the CNN algorithm achieves the highest accuracy of 90%, with rapid convergence starting from round 6. Conversely, the GRU algorithm produces fluctuating results throughout each round, indicating poor learning ability for this L3 time-frame attribute set.

Empirical findings indicate that adopting the L3 time-frame attribute set achieves comparable results to using the complete attribute set of the data.

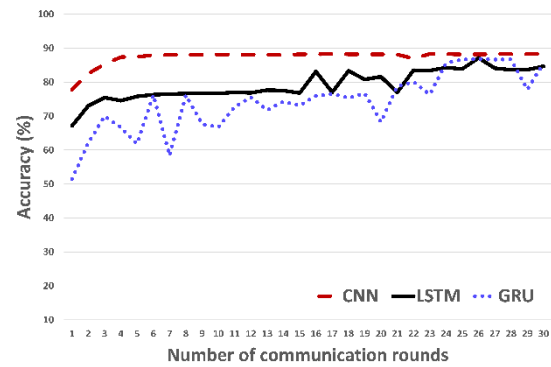


Fig. 17. Classification performance of the FL model based on class attribute L3 of the dataset

Table VI presents the evaluation metrics of the CNN, LSTM, and GRU algorithms when trained on centralized data using only the L5 time-frame attribute set, which collects information from a time frame of 5 seconds.

Based on the evaluation metrics presented in Table VI, the CNN, LSTM, and GRU algorithms were compared when trained on centralized data using only the L5 time-frame attribute set. The results indicate that the CNN algorithm achieved the highest accuracy of 87%, while the LSTM algorithm achieved an accuracy of 82%. However, the GRU algorithm gave relatively lower results with an accuracy of only 77%. It is noteworthy that the experimental results demonstrate that using only the L5 time-frame attributes produces quite similar results to using all the attributes of the data set.

TABLE VI
COMPARASION TABLE OF THE ALGORITHMS FOR CENTRALIZED
LEARNING BASED ON TIME FRAME (L5)

Metrics	CNN	LSTM	GRU
Precision	0.87	0.66	0.79
Recall	0.87	0.77	0.82
F1-score	0.83	0.70	0.79
Accuracy	0.87	0.77	0.82

The Fig 18 illustrates the classification results obtained by the FL model based on the L5 time-frame attribute set of the data set. The results indicate that the CNN algorithm provides the highest accuracy of 90% and exhibits the fastest convergence rate, starting from round 6. Conversely, the GRU algorithm's results exhibit significant fluctuations across each round, indicating that the model's learning ability for this L5 time-frame attribute set is insufficient. Based on the experimental results, it can be inferred that employing only the L5 time-frame attribute set yields results that are comparable to using all attributes of the dataset.

Based on the experiments conducted, the results indicate that the use of attribute sets based on time-frames yields comparable outcomes to using the entire dataset. Furthermore, a minor variation in accuracy was observed between centralized and distributed data models, fluctuating around 1%. However, the centralized approach is not adequate for ensuring data privacy, which is crucial in IoT networks. Therefore, the Federated Learning model is a more appropriate approach to preserving data privacy.

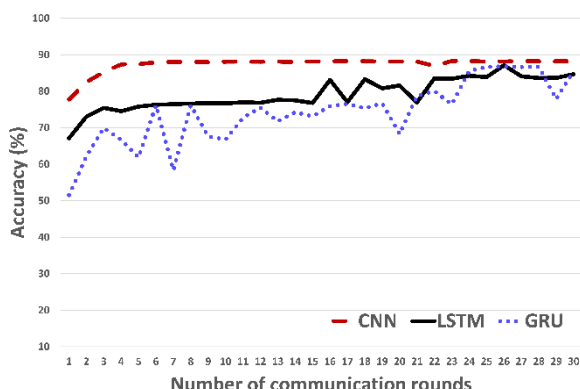


Fig. 18. Classification performance of the FL model based on class attributes L5 of the dataset

IV. CONCLUSION

Based on the results obtained from the N-BaIoT dataset, this study recommends using the Federated Learning (FL) approach for detecting IoT malware abnormal traffic while preserving user privacy. Two different methods were compared, i.e., the Federated Learning method, where each device owner trains a separate model, and the Centralized approach, where each device owner trains a single, isolated model using a centralized dataset. Although the Centralized approach still has slightly higher accuracy in detecting abnormal IoT traffic than FL, the difference is insignificant. However, the Centralized approach model fails to ensure the security and privacy of confidential information, unlike the FL model. Furthermore, when comparing the CNN, LSTM, and GRU models applied in the FL approach, CNN yields the best results and is suitable for simple FL models.

In future studies, our research aims to further optimize the Federated Learning model, specifically focusing on the global parameter to enhance its accuracy in detecting abnormal IoT traffic. Additionally, we plan to improve the processing time to achieve more efficient and effective results.

ACKNOWLEDGMENT

The study was financially supported by the Russian Science Foundation within of scientific project No. 22-49-02023 "Development and study of methods for obtaining the reliability of tethered high-altitude unmanned telecommunication platforms of a new generation.

REFERENCES

- [1] Fan, Xiaochen, et al. "BuildSenSys: Reusing building sensing data for traffic prediction with cross-domain learning." *IEEE Transactions on Mobile Computing* 20.6 (2020): 2154-2171.
- [2] Alrawi, Omar, et al. "Sok: Security evaluation of home-based iot deployments." *2019 IEEE symposium on security and privacy (sp)*. IEEE, 2019.
- [3] Kulik, V., & Kirichek, R. (2018, November). The heterogeneous gateways in the industrial internet of things. In *2018 10th International congress on ultra modern telecommunications and control systems and workshops (ICUMT)* (pp. 1-5). IEEE.
- [4] Kumar, J. Sathish, and Dhiren R. Patel. "A survey on internet of things: Security and privacy issues." *International Journal of Computer Applications* 90.11 (2014).
- [5] Butun, Ismail, Patrik Österberg, and Houbing Song. "Security of the Internet of Things: Vulnerabilities, attacks, and countermeasures." *IEEE Communications Surveys & Tutorials* 22.1 (2019): 616-644.
- [6] Radhakrishnan, Divya. "Internet of things: Privacy and security issues: A qualitative study about the internet of things and its privacy and security challenges from a developer's perspective." (2021).
- [7] Tsimenidis, S., Lagkas, T., & Rantos, K. (2022). Deep learning in IoT intrusion detection. *Journal of network and systems management*, 30, 1-40.
- [8] Aldweesh, A., Derhab, A., & Emam, A. Z. (2020). Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, 189, 105124.
- [9] Kairouz, Peter, et al. "Advances and open problems in federated learning." *Foundations and Trends® in Machine Learning* 14.1-2 (2021): 1-210.
- [10] Otoum, Safa, Ismaeel Al Ridhawi, and Hussein T. Mouftah. "Blockchain-supported federated learning for trustworthy vehicular networks." *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020.
- [11] Meidan, Yair, et al. "N-baiot—network-based detection of iot botnet attacks using deep autoencoders." *IEEE Pervasive Computing* 17.3 (2018): 12-22.
- [12] Arbex, Gustavo Vitral, et al. "IoT DDoS Detection Based on Stream Learning." *2021 12th International Conference on Network of the Future (NoF)*. IEEE, 2021.
- [13] Ferdowsi, Aidin, and Walid Saad. "Generative adversarial networks for distributed intrusion detection in the internet of things." *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019.
- [14] Almiani, Muder, et al. "Deep recurrent neural network for IoT intrusion detection system." *Simulation Modelling Practice and Theory* 101 (2020): 102031.
- [15] Xia, Qinyu, Shi Dong, and Tao Peng. "An Abnormal Traffic Detection Method for IoT Devices Based on Federated Learning and Depthwise Separable Convolutional Neural Networks." *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. IEEE, 2022.
- [16] Sim, Khe Chai, et al. "Domain Adaptation Using Factorized Hidden Layer for Robust Automatic Speech Recognition." *Interspeech*. 2018.
- [17] Almiani, Muder, et al. "Deep recurrent neural network for IoT intrusion detection system." *Simulation Modelling Practice and Theory* 101 (2020): 102031.
- [18] Vladimirov, Sergey, and Ruslan Kirichek. "The IoT identification procedure based on the degraded flash memory sector." *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*. Springer, Cham, 2017. 66-74.
- [19] Lou, Yang, et al. "Predicting network controllability robustness: A convolutional neural network approach." *IEEE Transactions on Cybernetics* 52.5 (2020): 4052-4063.
- [20] Amanullah, Mohamed Ahzam, et al. "Deep learning and big data technologies for IoT security." *Computer Communications* 151 (2020): 495-517.
- [21] Sim, Khe Chai, et al. "Domain Adaptation Using Factorized Hidden Layer for Robust Automatic Speech Recognition." *Interspeech*. 2018.
- [22] Siniosoglou, Ilias, et al. "A unified deep learning anomaly detection and classification approach for smart grid environments." *IEEE Transactions on Network and Service Management* 18.2 (2021): 1137-1151.
- [23] Larriba-Novoa, Xavier, et al. "An IoT-focused intrusion detection system approach based on preprocessing characterization for cybersecurity datasets." *Sensors* 21.2 (2021): 656.
- [24] Zhu, Hangyu, Haoyu Zhang, and Yaochu Jin. "From federated learning to federated neural architecture search: a survey." *Complex & Intelligent Systems* 7 (2021): 639-657.
- [25] Abbasi, Mahmoud, Amir Taherkordi, and Amin Shahraki. "FLITC: A Novel Federated Learning-Based Method for IoT Traffic Classification." *2022 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2022.
- [26] Mun, Hyunsu, and Youngseok Lee. "Internet traffic classification with federated learning." *Electronics* 10.1 (2020): 27.
- [27] He, Zhimin, et al. "Edge device identification based on federated learning and network traffic feature engineering." *IEEE Transactions on Cognitive Communications and Networking* 8.4 (2021): 1898-1909.
- [28] Wen, Jie, et al. "A survey on federated learning: challenges and applications." *International Journal of Machine Learning and Cybernetics* (2022): 1-23.
- [29] Zhu, Hangyu, Haoyu Zhang, and Yaochu Jin. "From federated learning to federated neural architecture search: a survey." *Complex & Intelligent Systems* 7 (2021): 639-657.
- [30] Do, Phuc Hao, et al. "An Efficient Feature Extraction Method for Attack Classification in IoT Networks." *2021 13th International*

Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). IEEE, 2021.

- [31] Cohen, Patricia, Stephen G. West, and Leona S. Aiken. Applied multiple regression/correlation analysis for the behavioral sciences. *Psychology press*, 2014.
- [32] Conrod, Patricia J., et al. "Effectiveness of a selective, personality-targeted prevention program for adolescent alcohol use and misuse: a cluster randomized controlled trial." *JAMA psychiatry* 70.3 (2013): 334-342.
- [33] Cerda, Patricio, Gaël Varoquaux, and Balázs Kégl. "Similarity encoding for learning with dirty categorical variables." *Machine Learning* 107.8 (2018): 1477-1494.
- [34] Guo, Cheng, and Felix Berkhahn. "Entity embeddings of categorical variables." *arXiv preprint arXiv:1604.06737* (2016).
- [35] Zhou, Yuyang, et al. "Building an efficient intrusion detection system based on feature selection and ensemble classifier." *Computer networks* 174 (2020): 107247.
- [36] Mahfouz, Ahmed, et al. "Ensemble classifiers for network intrusion detection using a novel network attack dataset." *Future Internet* 12.11 (2020): 180.
- [37] Yamashita, Rikiya, et al. "Convolutional neural networks: an overview and application in radiology." *Insights into imaging* 9 (2018): 611-629.
- [38] Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555* (2014).
- [39] Volkov, A., Khakimov, A., Muthanna, A., Kirichek, R., Vladyko, A., & Koucheryavy, A. (2017, June). Interaction of the IoT traffic generated by a smart city segment with SDN core network. *In International Conference on Wired/Wireless Internet Communication* (pp. 115-126). Springer, Cham.



Phuc Hao Do received his MS degree in Computer science from the University of Danang - University of Science and Technology in 2017. He is currently a Ph.D. student in the Department of Communication Networks and Data Transmission at the Bonch-Bruevich Saint- Petersburg State University of Telecommunications, Russia. His research interests include AI, ML, D and its application in different fields like network, blockchain.

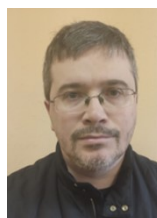


Dr. Tran Duc Le acquired his degree of Ph.D. at Admiral Makarov State University of Maritime and Inland Shipping, Russia in 2018. He has been working in Information Technology Faculty, The University of Danang - University of Science and Technology, Danang, Vietnam since 2019. His research areas include the Internet of Things, Security Analytics, Malware Analysis.



Dr. Sc. Vladimir M. Vishnevsky received the Engineering degree in applied mathematics from the Moscow Institute of Electronics and Mathematics, Russia, in 1971, the Ph.D. degree in queuing theory and telecommunication networks and the D.Sc. degree in telecommunication networks from the V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences (ICS RAS), in 1974 and 1988, respectively. He became a Full Professor with ICS

RAS in 1989 and the Moscow Institute of Physics and Technology in 1990. He was an Assistant Head of the Institute of Information Transmission Problems of RAS from 1990 to 2010 and an Assistant Head of laboratory with ICS RAS from 1971 to 1990. He is currently the Head of Telecommunication Networks Laboratory, ICS RAS. He is a member of Expert Councils of Russian High Certifying Commission and Russian Foundation for Basic Research, member of IEEE Communication Society, International Telecommunications Academy and New York Academy of Science. He has authored over 300 papers in queuing theory and telecommunications. He is a Co-Chair of IEEE conferences - ICUMT, RTUWO, and the General Chair of DCCN conference. His research interests lie in the areas of computer systems and networks, queuing systems, telecommunications, discrete mathematics (extremal graph theory, mathematical programming) and wireless information transmission networks.



Programming Engineering and Computer Science.

Dr. Aleksandr Berezkin, is working at the Bonch Bruevich Saint Petersburg State University of Telecommunications as the Associate Professor of Department of Programming Engineering and Computer Science. Science interest are Computer Vision and Machine Learning. In 2009, he defended his thesis with the topic "Model and method of decoding error correction based on neural network". Now he is doctoral student at the Department of



Petersburg State University of Telecommunications in 2012 and Dr.Sc. at the Povolzhskiy State University of Telecommunications and Informatics in 2018. From 2008 to 2013 he worked as a senior researcher at the Federal State Unitary Enterprise "Center-Inform". Since 2012 he has been working as the Head of the Internet of Things Laboratory at the Bonch-Bruevich Saint Petersburg State University of Telecommunications. Since 2017 he has been working as ITU-T Q12/11 Rapporteur in "Testing of Internet of things, its applications and identification systems". Since 2023 he has been working as the Rector of the Bonch-Bruevich Saint Petersburg State University. He is a General Chair of the International Conference "Internet of Things and Its Enablers" (inthiten.org).

Dr. Sc. Ruslan Kirichek is working at the Bonch Bruevich Saint Petersburg State University of Telecommunications as the head of Department of Programming Engineering and Computer Science. He was born in 1982 in Tartu (Estonia). He graduated Military-Space Academy A.F. Mozhaiskogo and the Bonch-Bruevich St. Petersburg State University of Telecommunications in 2004 and 2007, respectively. He received Ph.D. at the Bonch-Bruevich St.