





## Smart Intrusion Detection in IoT Edge Computing Using Federated Learning

Samir Fenanir<sup>1\*</sup>, Fouzi Semchedine<sup>2</sup>

<sup>1</sup> Department of Computer Science, Faculty of Sciences, University of Sétif 1, Sétif 19000, Algeria

<sup>2</sup> Mechatronics Laboratory, Optics and Precision Mechanics Institute, University of Sétif 1, Sétif 19000, Algeria

Corresponding Author Email: [samir.fenanir@univ-setif.dz](mailto:samir.fenanir@univ-setif.dz)

<https://doi.org/10.18280/ria.370505>

**Received:** 30 June 2023

**Revised:** 1 September 2023

**Accepted:** 9 September 2023

**Available online:** 31 October 2023

### **Keywords:**

*Internet of Things (IoT), smart intrusion detection (SID), deep learning, federated learning (FL), Edge Computing, fog computing*

### **ABSTRACT**

With the proliferation of the Internet of Things (IoT) in various domains, concerns over information security and user privacy have exponentially escalated. Numerous smart intrusion detection (SID) strategies, primarily based on machine/deep learning techniques, have been proposed to counter these security challenges. However, these strategies are typically designed with a centralized approach, where IoT devices relay their data to a central server for training, potentially exposing the data to a range of security threats and privacy vulnerabilities. To address these data security and privacy challenges, a federated learning (FL) approach is adopted in this study. In this approach, individual users train their local models and transmit only parameter updates to the server. These parameters are then aggregated to form the global model. In each FL training cycle, IoT users receive an updated global model from the central server, which they further train utilizing their respective local datasets. This methodology allows for the preservation of IoT device privacy while optimizing the global model. In the context of IoT edge computing, where computational load is distributed to network edges for efficient resource utilization, a novel SID approach based on federated learning is proposed. The effectiveness of this approach is evaluated using three popular deep learning models and three well-established IoT field datasets. This thorough evaluation serves to assess the generalizability of the models and validate the reliability of the results. Through extensive experiments and comprehensive comparisons with other methodologies, this study demonstrates superior performance, achieving an impressive 99% accuracy rate. This result underscores the robustness of the proposed approach in accurately detecting intrusions within IoT environments, thereby offering a promising solution for securing IoT edge computing.

## **1. INTRODUCTION**

The Internet of Things (IoT) is a combination of a variety of connected objects using all kinds of communication technologies, forming ubiquitous computing [1]. The IoT is the integration of the Internet with the physical world by harnessing Artificial Intelligence (AI) to deliver smarter services to the environment [2]. Nowadays, the IoT provides an increasing number of applications and services in various fields such as: domestic, education, agriculture, energy distribution, health, tourism, transport, etc.

Though the IoT offers many benefits, it also faces many difficulties and challenges [3]. Among the major challenges of the IoT is to manage a multitude of heterogeneous objects connected by heterogeneous communication technologies to a variety of applications and users [4]. Added to this is the open nature of the IoT and the presence of components with low computing capabilities and power, which makes it vulnerable to various types of attacks, and most existing security solutions become difficult to apply [5]. On the other hand, information confidentiality and user privacy are also vulnerable. For example, in medical applications, many internet-connected sensors embedded into the human body for daily use reveal their habits, state of health, geographic location and other types of information. Therefore, this critical information must be carefully secured from its acquisition to its management

and use [6]. Consequently, IoT security and privacy have become a fundamental problem and a major concern.

To this end, Intrusion Detection System (IDS) is widely used as a security mechanism to detect different kinds of attacks on the IoT ecosystem. Depending on the detection methods used, IDSs are divided into three main categories: signature-based detection, anomaly-based detection, and specification-based detection [7]. Signature-based IDS detects attacks by comparing its signatures to predefined attack models that are already stored in a database [8]. This technique is straightforward to employ. Nonetheless, it comes with a high cost, especially as the number of attacks grows, requiring additional storage space. Moreover, its primary drawback lies in its limited scope, as it can only detect attacks that match the existing signatures. Consequently, there is a constant need for database updates to incorporate new attack signatures. Anomaly-based IDS detects new intrusions by comparing new entries to its pattern of normal behavior. Any deviation exceeding a predefined threshold is marked as an anomaly [9]. The primary advantage of this technique is its ability to identify new attacks by flagging any deviations from normal behavior. Nevertheless, it frequently produces numerous false positives because not every deviation from normal behavior indicates an actual attack. Specification-based IDS is a hybrid method that combines the two previous techniques. This technique leverages both techniques to identify new attacks

while also reducing false positives. However, implementing such mechanisms consumes additional energy and resources. In our study, we adopted anomaly-based SID to detect and classify different kinds of attacks using deep learning techniques, which have proven their efficiency in the security domain.

Learning techniques for the IoT can be classified into three categories, namely centralized, distributed and federated learning [10]. In the centralized approach, attacks can be detected with high accuracy. However, the data transfer cost from the IoT network to the server and the latency are high in this model due to the large distance between IoT devices and the central server. Moreover, this approach does not guarantee the confidentiality of user data. Indeed, the sharing of sensitive data on the IoT network will make them vulnerable to various attacks. Although distributed learning solves the latency problem by processing data close to the IoT devices, it also does not guarantee data privacy. Federated learning is a new paradigm to address the limitations of centralized and distributed learning [11], it has been widely used in recent years to guarantee the confidentiality of user data and to reduce latency.

Federated learning (FL) for the IoT is essentially composed by two components: users and aggregator. Users train on local models and then send update parameters to the aggregator which aggregates them to generate a new global model. Generally, two approaches are commonly used: the edge-based approach and the cloud-based approach for learning model global. In edge-based FL, global model aggregation is done on the edge server, while in cloud-based FL, it is done on the cloud server. Given the many users located in a large distributed area. Cloud-based FL is better suited for training more general models than edge-based FL [12].

In this paper, we propose smart intrusion detection based on federated learning for IoT network, using three deep learning algorithms, namely DNN, CNN, and LSTM. To evaluate our approach, we used three datasets, namely IoTID20, IoT-23 and N-BaIoT, which allow us to compare the different models and select the best model for the IoT network. The contributions of our work are as follows:

- We proposed a novel approach based on federated learning for intrusion detection in IoT systems. This approach addresses the data security and privacy concerns associated with centralized approaches.
- We conducted experiments using three deep learning models and three popular datasets in the IoT domain to evaluate the effectiveness and generalization capability of our approach.
- Through rigorous evaluation and comparisons with existing approaches, we demonstrated superior performance in terms of accuracy, precision, recall, and F-score.
- Our approach proved to be efficient, reliable, and capable of effectively detecting intrusions in edge IoT systems.
- The results of our experiments highlight the effectiveness and potential of federated learning in the context of intrusion detection for IoT systems.

Finally, our work contributes to advancing the field of IoT security by introducing a privacy-preserving approach that achieves high detection accuracy and addresses the challenges of centralized learning approaches.

The remaining of this paper is organized as follows: Section 2 presents related work. Section 3 provides an overview of learning techniques. Section 4 details the proposed approach. Section 5 presents the data set, evaluation metrics and the

proposed approach results. Finally, Section 6 concludes the paper.

## 2. RELATED WORK

Recently, several IDS works using machine/deep learning techniques have been developed to provide better protection for the IoT ecosystem. In this section, we discuss some works by classifying them into two categories:

### 2.1 Machine Learning-based approach

Fenanir et al. [5] proposed a lightweight IDS based on an elaborate combination of feature selection based on correlation and classification techniques. A comparison of seven classification algorithms were performed on 3 datasets, namely KDD-99, NSL-KDD and UNSW-NB15 datasets. Finally, the decision tree algorithm was selected owing to its performance on several datasets.

Moustafa et al. [13] suggested an IDS to identify malicious events in IoT networks, based on analysis of DNS flows. To evaluate this technique, an Adaptive Boosting ensemble learning approach is developed using three algorithms: Decision Tree, Naive Bayesian and Neural Network. Experimental results using UNSW-NB15 and NIMS botnet datasets show that the proposed framework provides good performance.

Verma and Ranga [14] reports a benchmarking study on IDS using ML classifiers for IoT networks. The classifiers were evaluated on the CIDDS-001, UNSWNB15, and NSL-KDD datasets. The experimental results show good performance, in particular with the regression trees and the gradient boosting classifier.

Kumar et al. [15] proposed an IDS based on the design of distributed ensemble using Fog computing. This approach combines in the first level, three classifiers, namely K-Nearest Neighbors (KNN), eXtreme Gradient Boosting (XGBoost) and Naive Bayesian (NB). At the second level, the random forest is used for the final classification. UNSW-NB15 and DS2OS datasets are used to validate the proposed approach. The experimental result shows that the proposed approach provides a higher detection rate, especially on the DS2OS dataset, where the detection rate can reach 99.99% for most attack categories.

Pajouh et al. [16] proposed an IDS in IoT Backbone Networks. This model used two dimension reduction techniques: Principal component analysis and linear discriminant analysis, and two classification algorithms: K-Nearest Neighbors and Naïve Bayesian. The proposed approach is designed to detect the low frequency attacks using the NSL-KDD dataset and two types of attacks: User-to-Root and Remote-to-Local attacks. The experiment results of this model provide an 84.66% detection rate for binary classification.

Alruhaily and Ibrahim [17] presented a multi-layer IDS for WSN, which uses two layers of detection. The first layer uses the Naive Bayes algorithm to detect suspicious packets at the network edge. The second layer uses a Random Forest multiclass classifier to deeply analyze the inspected packets located at the cloud. The WSN-DS dataset is used to validate the proposed approach. The experiment results give a relatively high performance with different evaluation metrics. However, the proposed model did not use any feature selection

techniques to reduce data dimensionality. Moreover, it uses a unique database, which does not allow to effectively evaluate the system.

## 2.2 Deep Learning-based approach

Almogren [18] proposed an IDS to detect intrusions in the Edge of things (EoT) network based on the deep belief network (DBN). The evaluation of this model is performed using the UNSW-NB15 dataset with various DBN structures. The experiment results indicate that DBN has the best overall performance compared to ANN and SVM algorithms, where the detection rate has reached (96.34%). Although the deep belief-based approaches give high accuracy, they are not efficient in terms of inference time.

Li et al. [19] presented an IDS based on an industrial IoT network using a multi-convolutional neural network. This model was evaluated using the unique KDD dataset and provided 86.95% accuracy for binary classification and 81.33% accuracy for classification multiclass. However, this study was limited to only one benchmark dataset, which affects the

ability to generalize results.

Li et al. [20] adopted a disagreement learning based IDS in real IoT network environments. This method aims to improve the detection rate and reduce false alarms. The results show good performance using unlabeled datasets. However, its efficiency is uncertain with large samples. Moreover, the dataset should be updated regularly to keep the learning efficiency.

Abdalgawad et al. [21] implemented two adversarial generative deep learning methods, namely Autoencoders and Bidirectional GANs to detect attacks in IoT environment. The proposed model used the IoT-23 dataset. The experimental results showed that the model performs well with an F1 rate of up to 0.99. While this model was evaluated using different metrics, inference time was not evaluated.

Fenanir et al. [3] proposed a centralized IDS for IoT network, using a semi-supervised learning autoencoder. This model was evaluated using the NSL-KDD and CIDDS-001 datasets. The authors adopted a centralized architecture which is less suited to the IoT network given its distributed nature.

**Table 1.** Summary of related work

Ref.	Datasets	Classifiers	Key Findings
[3]	NSL-KDD, CIDDS-001	Deep AutoEncoder	This paper proposed a centralized IDS for IoT network, using a semi-supervised learning autoencoder. The authors adopted a centralized architecture which is less suited to the IoT network given its distributed nature.
[5]	KDD-99, NSL-KDD, and UNSW-NB15	KNN, SVM, Decision Tree, Random Forest, Regression Logistic, Naive Bayes, MLP	This paper proposed a lightweight IDS that combined feature selection and classification techniques. The decision tree algorithm (DT) is chosen as the optimal model.
[13]	UNSW-NB15, NIMS botnet	Decision Tree, Naive Bayesian, Neural Network classifiers	The authors demonstrate the effectiveness of their approach through an experimental evaluation using an ensemble intrusion detection technique based on DNS flow analysis to protect Internet of Things (IoT) network traffic.
[14]	CIDDS-001, UNSWNB15, NSL-KDD	regression trees and gradient boosting	The authors conducted a benchmarking study on IDS for IoT networks, highlighting the effectiveness of regression trees and gradient boosting classifiers on datasets.
[15]	DS2OS	K-NN, Gradient Boosting, Naive Bayesian, random forest.	This paper proposed a distributed ensemble IDS using Fog computing, with a combination of classifiers such as K-Nearest Neighbors, eXtreme Gradient Boosting, Naive Bayesian, and random forest.
[16]	NSL-KDD	KNN, Naive Bayesian	The authors developed an IDS for IoT Backbone Networks, utilizing Principal Component Analysis and Linear Discriminant Analysis for dimension reduction and K-Nearest Neighbors and Naive Bayesian classifiers for low-frequency attack detection.
[17]	WSN-DS	Naive Bayes and Random Forest	This paper presented a multi-layer IDS for WSN, employing Naive Bayes and Random Forest classifiers for suspicious packet detection. While achieving high performance, their model lacked feature selection techniques and relied on a unique database for evaluation.
[18]	UNSW-NB15	deep belief networks (DBN)	The authors introduced an IDS for Edge of Things (EoT) networks based on deep belief networks (DBN). The DBN outperformed ANN and SVM algorithms, However, DBN models were not efficient in terms of inference time.
[19]	KDD	convolutional neural network (CNN)	The authors developed an IDS for industrial IoT networks using a multi-convolutional neural network. The study was limited to a single benchmark dataset, impacting generalizability.
[20]	Generic Dataset	disagreement-based Method	This paper proposed a disagreement learning-based IDS for real IoT networks, demonstrating good performance with unlabeled datasets. However, its efficiency with large samples was uncertain, and regular dataset updates were necessary for learning efficiency.
[21]	IoT-23	Autoencoders and Bidirectional GANs	The authors implemented adversarial generative deep learning methods for IoT attack detection using the IoT-23 dataset. The model achieved a high F1 score of up to 0.99, although inference time was not evaluated.
[22]	-	-	The authors reviewed the application of Federated Learning in the context of IDS and highlight the challenges associated with its implementation. The study provides insights into the potential benefits and limitations of using Federated Learning for IoT intrusion detection, contributing to the advancement of this field.

Campos et al. [22] conducted a comprehensive review on the utilization of Federated Learning for intrusion detection in the Internet of Things (IoT) domain. They specifically examined the application of Federated Learning in this context and shed light on the implementation challenges involved. The study offers valuable insights into the potential advantages and drawbacks of employing Federated Learning for IoT intrusion detection, thereby making a significant contribution to the advancement of this field.

In Table 1, we have compiled a summary of the main contributions from similar studies. The table includes information such as experiment dataset, classifiers used, and key findings.

Based on previous studies and considering its limitations, we proposed a federated learning approach for smart intrusion detection in IoT environment. This approach entails training models at the edge of IoT devices and consolidating their updated parameters at a central server. This method ensures the protection of IoT device privacy while simultaneously enhancing the performance of the overall model. We implemented three deep learning models, namely DNN, CNN, and LSTM, to conduct a comparative analysis and select the most suitable classifiers. To evaluate our approach and assess the generalizability of the models, we used three commonly employed datasets in the IoT field: IoTID20, IoT23, and N-BaIoT datasets.

### 3. BACKGROUND

In this section, we present a brief introduction to cloud, fog and Edge computing paradigm, followed by an overview of three ML architectures namely, centralized, distributed, and federated learning.

#### 3.1 Cloud computing

Cloud computing is an infrastructure that provides a variety of services for storing, gathering and processing data using remote servers hosted on the Internet. The main services offered by cloud computing are: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

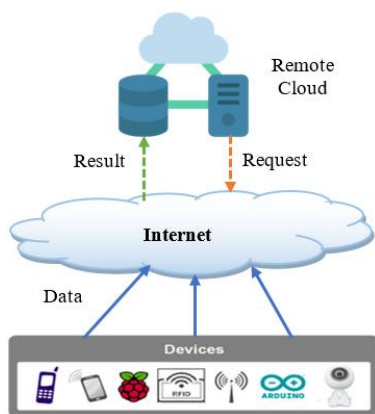


Figure 1. Conventional cloud computing in IoT

The integration of cloud computing in the IoT provides a new data storage area and services for processing, analyzing and storing a huge volume of data generated by the IoT [23]. The primary goal of cloud computing in IoT is to increase

efficiency and performance, as well as enable resource sharing [24]. However, cloud computing in IoT faces certain communication limitations, particularly in terms of bandwidth and latency constraints. This is where edge computing becomes advantageous, especially in remote locations, as it enables data processing and analysis to be performed closer to the source of data generation [25].

Figure 1 shows the conventional cloud computing in IoT, where IoT devices generate data and transfer it to the remote cloud via the Internet. The remote cloud then sends a data consumption request and receives the result.

#### 3.2 Edge computing

While data processing in the cloud offers greater computational power compared to the edge, the increasing amount of data generated by the IoT requires a significant bandwidth and leads to high latency, which makes cloud computing inefficient to manage all this data [26]. In edge computing, data processing occurs in data sources that are close to end users, providing adequate computing power to handle IoT requirements [18]. Thus, there is no need for the IoT nodes to send data to the cloud. Instead, the data can be processed directly on the nodes, leading to a significant reduction in latency.

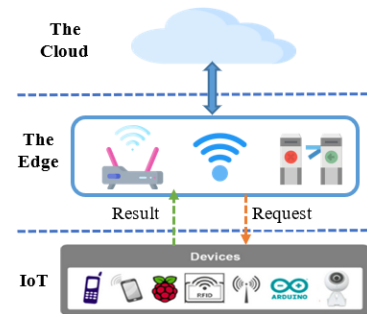


Figure 2. Edge computing in IoT

Figure 2 depicts edge computing in IoT. At the IoT network edge, data from devices is received and analyzed before being sent to a data center or cloud.

#### 3.3 Fog computing

Fog Computing refers to a decentralized infrastructure, which transfers data and services between the cloud and the network edge [27]. It addresses the challenges of latency, bandwidth consumption, and communication optimization between IoT devices and remote cloud services by leveraging proximity storage and processing capabilities [28]. By bringing computing resources closer to the data source, fog computing reduces the reliance on distant cloud servers and enables real-time data processing and decision-making. This proximity-based approach improves the overall responsiveness and reliability of IoT applications. Fog computing supports the mobility, scalability, and high availability requirements of IoT systems, accommodating the dynamic nature of IoT deployments. Furthermore, fog computing plays a crucial role in managing the heterogeneity of IoT devices and applications. It achieves this through the support of interoperability mechanisms and the use of application programming interfaces (APIs) that facilitate seamless integration and communication between different

IoT platforms and protocols. Virtualization technologies are also employed in fog computing to enable the efficient allocation and utilization of resources across diverse IoT devices and services [27].

In summary, fog computing offers a decentralized infrastructure that optimizes communication, reduces latency, and enhances the scalability and interoperability of IoT applications. By leveraging proximity-based storage and processing, fog computing brings computational capabilities closer to the network edge, enabling efficient and reliable data processing for a wide range of IoT use cases.

Figure 3 shows a reference architecture for fog computing. At the lowest level are the IoT devices, which collect data from the environment and transmit them to the edge layer. The edge layer includes edge nodes that serve as a gateway to the fog layer. In the fog layer, there are multiple fog nodes with some compute and storage capability, which allow them to manage, connect, and share resources between the edge layer and the cloud layer [29]. The next layer consists of cloud services and resources that manage resources and process received IoT tasks. The top layer contains IoT applications that harness fog layer to offer innovative and smart solutions to end users.



Figure 3. Fog computing architecture

### 3.4 Centralized learning

Centralized learning for IoT requires transmitting training data from each IoT device to the cloud server to create a common model [30]. This model will be used by all user devices. The main advantage of this centralized learning is the ability to generalize from a subset of devices and thus perform with other compatible devices. Moreover, the centralized learning is computationally efficient because the IoT devices are free from intensive computational work that requires high computational resources [3]. However, the centralized approach applied to the IoT presents difficult challenges. In such an approach, the bandwidth is often very limited and the data amount sent to the cloud is very large, since training of complex tasks usually requires exchanging large blocks of data to a central server. Moreover, when there is a lot of interaction with services available in the cloud, latency due to the round trip to the cloud is often a problem

for applications running in real time. Additionally, energy consumption is a significant concern in this architecture, as it can lead to faster battery drain for IoT devices. Frequent data transmission to the central server and back can also strain network resources. Furthermore, the data exchanged by users can be very private, such as personal identification information, payment data, protected health information, etc. When these private data are shared on the cloud, there is a high possibility that user privacy will be compromised by eavesdropping attacks [30]. The development trends of this approach include: Advancements in Deep Learning, particularly with the emergence of larger and more efficient neural networks. Additionally, techniques for compressing and optimizing models for deployment on edge devices have gained prominence, resulting in reduced computational and energy requirements for centralized learning. This approach is presented in subgraph (a) of Figure 4.

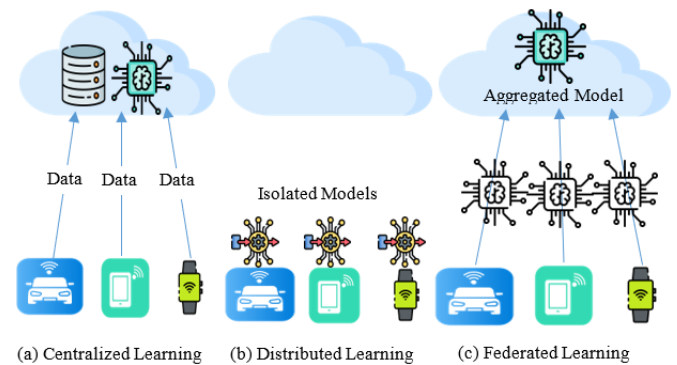


Figure 4. Machine learning approaches [22]

### 3.5 Distributed learning

Distributed learning techniques are exploited to solve complex algorithmic problems on large-scale data sets by assigning the learning task to distributed devices rather than a central server [31]. As shown in subgraph (b) of Figure 4, each device learns an individual model of its environment independently of other devices. The advantages of this technique are that the model adapts to changes as they happen, learning is not limited by internet connection, and no confidential information should be transferred to the cloud. However, distributed learning applied to IoT is constrained by the limited computing capacity, high energy consumption and low battery capacity, which can hinder the effectiveness and efficiency of distributed learning algorithms. To overcome these limitations, it is crucial to develop lightweight learning models specifically designed for IoT environments [5]. These lightweight models aim to strike a balance between accuracy and resource consumption, ensuring that the learning process can be executed efficiently on resource-constrained IoT devices. Furthermore, edge devices have significantly enhanced their capabilities, enabling distributed learning models to autonomously tackle more complex tasks at the local level. Consequently, this diminishes the need for extensive data transmission and centralized processing.

### 3.6 Federated learning

Federated learning also relies on shifting the training task to the IoT devices, and then federating local models and learning on the central cloud server. This approach combines

the benefits of the two previous approaches, as presented in subgraph (c) of Figure 4. The central cloud server distributes a generic learning model to the devices. Each device independently trains the model using its own local data and transmits weight updates to the server. Finally, the server calculates the average and aggregates the global model, which the devices will use in subsequent training cycles. The process will repeat until the desired level of convergence is achieved [32]. This architecture can reduce energy consumption compared to pure centralized approaches because it minimizes the need for frequent, data-intensive transmissions. It allows IoT devices to train locally and communicate selectively. Due to its advantages over competing approaches, federated learning in the IoT has garnered a lot of attention lately. However, there are still some challenges to overcome, communication issues, heterogeneity, and privacy invasions [33].

#### 4. PROPOSED METHOD

Among the main challenges we faced was to create a lightweight SID that adapts to the limited processing power of the IoT devices, respects user privacy, and reduces latency and communication costs. Traditional centralized learning methods transfer all local datasets to a remote server for training, which solves the problem of limited capacity of IoT devices [3, 5]. However, this can expose the data to many types of attacks without ensuring the protection of users' privacy and preventing data leakage. To address these concerns, we propose a new federated learning based-approach to detect intrusions in IoT networks. The benefits of this approach is to maximize the learning efficiency by spreading the computations and learning over multiple local data in the IoT network. It also ensures data sensitivity and privacy by protecting device information and simply distributing local model updates.

##### 4.1 Proposed model architecture

The proposed approach architecture, shown in Figure 5, comprises three layers. The application layer contains IoT applications and services that serve to manage resources and process tasks to provide innovative and intelligent solutions to end users. The next layer includes IoT devices, which collect information from the environment for training local models. It consists of n users, each user having a local model of a single device. At the upper layer there is a central server that orchestrates the federated learning process and detects intrusions.

The federated learning process consists of three essential steps: training local models, aggregating global models and updating local models. The steps of this process are as follows:

Step 1: Initially, the central server in the cloud layer initializes the global model by pre-training it with a set of initial weights. Subsequently, the global model is distributed to the individual users. It is worth noting that certain studies have addressed the aspect of user selection, wherein a subset of users is chosen based on various criteria, such as device status (active/inactive), availability of an unlimited connection, battery level, and so on.

Step 2: Once the local models are obtained from the central server, each user in the edge layer trains its local model using

its training data. In the case of a concrete SID, the training process involves utilizing the local network traffic of each user.

Step 3: When the training is finished, the users send their updated model parameters/weights to the server. If a user exceeds the transfer time specified by the server due to a limited connection, insufficient computing power or a huge amount of training data, the update parameters of this user will be ignored and the process will continue with the updates received.

Step 4: The server aggregates all received parameters/weights to generate a new global model for the next training cycle. Various aggregation algorithms are employed in federated learning, including the original and widely-used Federated Averaging (FedAvg) [34], as well as more recent approaches like FedProx [35], which can be considered as an extension of FedAvg.

Step 5: The learning process, spanning from step 1 to step 5, constitutes a cycle that is iterated repeatedly until the desired level of precision or a predetermined number of cycles is reached.

Step 6: Upon completion of the learning process, the system becomes capable of detecting new attacks and issuing alerts in the event of intrusions.

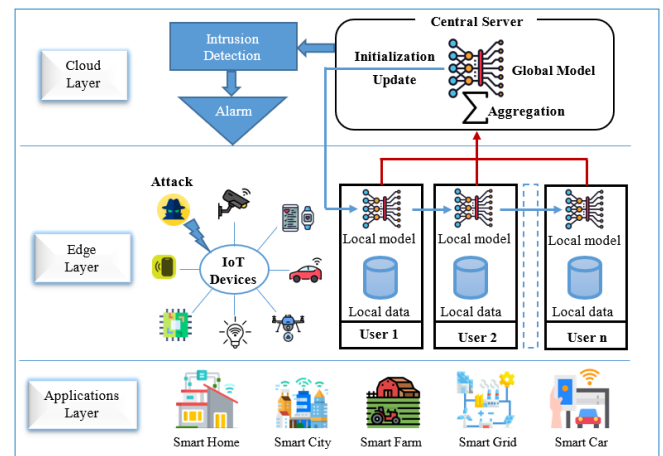


Figure 5. Architecture of the proposed approach

##### 4.2 Local learning models

Federated learning is performed with the number of available users, and each user has their own training data and local learning model. Choosing a less complex and more efficient local learning model presents a significant challenge and remains a major concern, as IoT devices are known for their limited computing capacity and power resources. Therefore, the local learning model should be adapted to the constraint related to IoT devices. To choose the most suitable local learning model that aligns with the limitations of IoT devices, we evaluated three deep learning algorithms, namely DNN, CNN and LSTM. Here is a brief description of each algorithm:

**Deep Neural Networks (DNN):** It consists of a set of neurons arranged in a multi-layered sequence, which allows data to be processed in a complex way, using advanced mathematical models.

**Convolutional Neural Networks (CNN):** It is composed of a set of conventional layers connected in sequence to generate an output from the analyzed input.

**Long Short-Term Memory (LSTM):** This type of Recurrent Neural Network (RNN) consists of interconnected units capable of learning and remembering long-term dependencies. It utilizes a memory structure that can retain its state over time.

### 4.3 Federated learning model

As described in section 4.1, the server coordinates with a set of users, and it aggregates the updates sent by the users in order to generate the global model which will be used in the next rounds. To illustrate how to train the FL model, assume a set of  $K$  users. Each user  $u_k$  ( $k \in [1, K]$ ) has an associated model weight  $w_k$ , and a local dataset  $d_k$  composed of  $s_k$  data samples. Each user updates the local model with their local data, using the following gradient descent formula:

$$w \leftarrow w - \eta \nabla \ell(w; b) \quad (1)$$

where,  $\eta$  is the rate of learning and  $b$  is the size of the local minibatch used for user updates. Then the server calculates the weighted average of the local models using the following formula:

$$w = \sum_{k=1}^K \frac{s_k}{S} w_k \quad (2)$$

where,  $S = \sum_{k=1}^K s_k$ .

We have employed an approach introduced by McMahan et al. [34], which utilizes the aggregate function FedAvg. The complete pseudo-code for this approach is presented in Algorithm 1.

---

#### Algorithm 1: Federated averaging [12]

---

##### Aggregation Server:

1. Weight initialization  $w^{t=0}$
  2. **for** every round  $t = 1, 2, \dots$  **do**
  3.    $m \leftarrow \max(C, K, 1)$
  4.    $N^t \leftarrow$  (Random set of  $m$  users)
  5.   **for** each user  $k \in N^t$  **in parallel do**
  6.      $w_k^t \leftarrow \text{UserUpdate}(k, w^{t-1})$
  7.   **end for**
  8.    $w^t \leftarrow \sum_{k=1}^{N^t} \frac{s_k}{S} w_k^t$
  9. **end for**
- UserUpdate(k, w):**
10.  $\beta \leftarrow$  *Splite  $d_i$  into batch of size  $B$*
  11. **for** every local epoch  $e$  **do**
  12.   **for** every batch  $b \in \beta$  **do**
  13.      $w \leftarrow w - \eta \nabla \ell(w; b)$
  14.   **end for**
  15. **end for**
  16. **return**  $w$
- 

### 4.4 Detection process

Algorithm 2 provides a detailed description of the intrusion detection procedure in our proposed approach. At the beginning ( $t=0$ ), the central server initializes the global model  $M$  and distributes it among all users. Each user has their own dataset  $d_k$ , which is spilled into a train set for local model training, and a test set for local model evaluation. Before starting the learning procedure, a pre-processing step

is performed to improve data quality by dealing with missing values, inconsistent values, duplicate instances, etc. This is followed by a feature selection step which aims at selecting the most relevant attributes of a dataset. This procedure increases storage efficiency, reduces computational costs, and enhance the performance of a ML model [3, 5].

---

#### Algorithm 2: The FL method for intrusion detection

---

1. **Input:** Local Datasets  $d_k$ ,
  2.   Global model  $M$
  3. **Output:** Final Intrusion Detection Model
  4. Send Global model  $M$  to all users
  5. **for** every user  $k \in N$  **do**
  6.   Splitting  $d_k$  into train and test dataset
  7.   Pre-processing and feature selection of  $d_k$
  8. **end for**
  9. **for** every round  $t = 1, 2, \dots$  **do**
  10.   **for** every user  $k \in N$  **do**
  11.     Training and testing on  $d_k$
  12.     Sending updated parameters to server
  13.   **end for**
  14.   **with** the server
  15.     Local Model Aggregation
  16.     Generating a new model global
  17.     Sending the model global to users
  18.   **end with**
  19. **end for**
  20. Evaluating Final Model
  21. **end algorithm**
- 

Once users receive the intrusion detection models from the server, each user learns a personalized local model and then returns only the update parameters to the server, instead of sharing sensitive information that could be vulnerable to theft. The server then aggregates the weights obtained from various user models to generate the current global model. This model is transmitted back to the users for the next round. This learning process is repeated until a certain number of rounds is completed or a specific level of precision is achieved. At the end, the server can identify various types of attacks and generate alert in the event of an intrusion.

## 5. EVALUATION RESULTS

In this section, we will present the results of the performance evaluation conducted on our proposed model. We will start by discussing the dataset that was utilized for our experiments, providing relevant details about its composition and characteristics. Following that, we will provide an overview of the performance metrics that were employed to assess the effectiveness of our model. Additionally, we will compare our approach with other related approaches to showcase its superiority and highlight its unique contributions. Finally, we will delve into the experimental results obtained, analyzing and interpreting the findings to provide a comprehensive understanding of the performance and efficacy of our proposed model.

### 5.1 Dataset

Before starting the evaluation of the model, the selection of a suitable dataset is crucial and requires careful consideration. To fulfill this objective, we employed three

datasets, namely IoTID20 [36], IoT-23 [37] and N-BaIoT [38] datasets, which improves the ability to generalize the results on the one hand and to effectively evaluate the system on the other.

**IoTID20 dataset:** This dataset was generated by connecting IoT devices to a smart home, resulting in 625,783 samples with 80 features. Among these samples, 585,710 are classified as malicious. Table 2 provides a description of the different label classes in this dataset.

**Table 2.** A description of the IoTID20 dataset

Label	Description	Sample Count
Normal	No suspicious or malicious activity	40,073
DoS	Denial of Service attacks	59,391
Mirai	Mirai botnet attacks	415,677
MITM	Man-In-TheMiddle attacks	35,377
Scan	Scan attacks	75,265
Total		625,783

**IoT-23 dataset:** This dataset is created by the Stratosphere Laboratory. It is based on the network traffic from IoT devices and consists of 20 malware captures and 3 benign captures. This dataset has 19 features and 325,307,990 samples of which 294,449,255 are malicious. A summary of this dataset is given in Table 3.

**Table 3.** A summary of the IoT-23 dataset

Label	Description	Sample Count
Attack	Some kind of infection attack between devices	9,398
Benign	No attack	30,858,735
C&C	A Command-and-Control Attack	21,995
DDoS	Distributed Denial of Service Attack	19,538,713
File Download	The infected device is downloading a file	71
Heart Beat	The C&C server monitors the infected host using the sent packets	34,518
Mirai	Mirai botnet attacks	2
Okiru	Okiru botnet attacks	60,990,711
Part of a Horizontal Port Scan	Scanning horizontal ports for information to launch additional attacks.	213,853,817
Torii	Torii botnet attacks	30
Total		325,307,990

**Table 4.** Sample count for the N-BaIoT dataset

Label	BENIGN	MIRAI	BASHLITE	Total
Device1	49,548	652,100	316,650	1,018,298
Device2	13,113	512,133	310,630	835,876
Device3	39,100	-	316,400	355,500
Device4	175,240	610,714	312,723	1,098,677
Device5	62,154	436,010	330,096	828,260
Device6	98,514	429,337	309,040	836,891
Device7	52,150	-	323,072	375,222
Device8	46,585	513,248	303,223	863,056
Device9	19,528	514,860	316,438	850,826

**N-BaIoT dataset:** The N-BaIoT dataset was collected from 9 IoT devices that were infected with 10 different types

of attacks. These attacks can be categorized into two groups: Mirai and BASHLITE. The dataset comprises 116 features and a total of 7,062,606 samples, out of which 6,506,674 samples are classified as malicious. The distribution of samples for each device is provided in Table 4.

## 5.2 Evaluation metrics

To evaluate the performance and compare the results of different models, various metrics are used, which are based on a confusion matrix consisting of four metric values: True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN).

The evaluation metrics employed are briefly defined below:

Accuracy is the rate of the true predictions. It is given by the formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

Precision is the number of successful positive predictions. It is calculated by the formula:

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

Recall is the percentage of positives that is well predicted. The calculation formula is as follows:

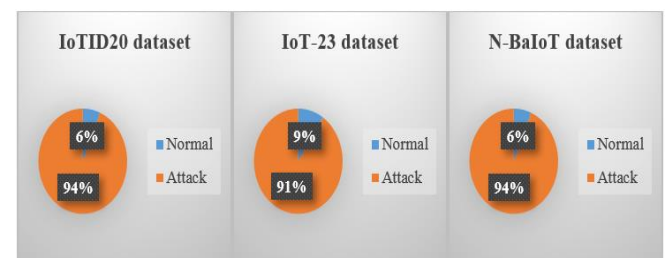
$$Recall = \frac{TP}{TP + FN} \quad (5)$$

F1-Score or F1-measure is the harmonic average of the precision and recall metrics. It is defined as follows:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (6)$$

## 5.3 Experimental results

In this section, we discuss the results obtained from various experiments conducted on the proposed model. All experiments were conducted with 9 users, where each user was assigned to one of the 9 devices. The N-BaIoT dataset consists of 9 devices, where each device is assigned to a single user. On the other hand, the IoT 23 and IoTID20 datasets were divided into 9 subsets to accommodate the user-device assignment. Figure 6 illustrates the distribution of classes in the three datasets.



**Figure 6.** Distribution of classes in the three datasets

We considered a binary classification problem with 2 classes: Normal and Attack, present in each dataset. It is evident that the class distribution is imbalanced, with the



attack class being predominant. This class imbalance may impact the performance of the models. However, despite this imbalance, these datasets are widely recognized benchmarks in recent studies, allowing for meaningful comparisons with similar research.

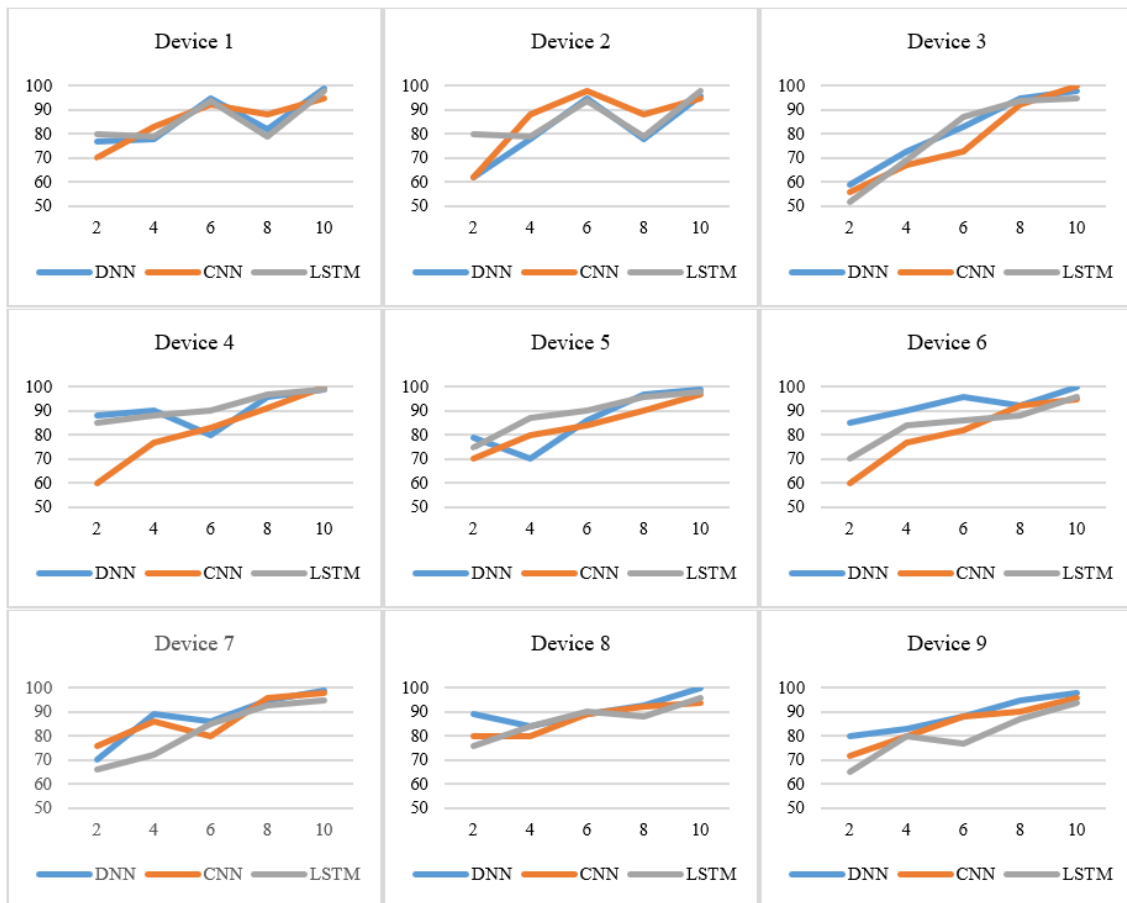
Before training the DNN, CNN, and LSTM classifiers, the datasets are preprocessed accordingly. The hyper-parameters used are: binary cross-entropy, ADAM optimizer and learning rate with a value of  $lr=0.001$ . The learning process is performed over 10 rounds. The implementation of the model is carried out using the Pytorch deep learning framework. For further reference, Table 5 provides the specific values of the key hyperparameters employed in our proposed approach for the various classifiers.

Figure 7 displays the accuracy trends in the 9 devices following each federated aggregation round. The DNN, CNN, and LSTM classifiers are applied to the N-BaIoT dataset for this analysis.

We conducted an evaluation of federated learning performance on the N-BaIoT dataset, involving 9 users in the model training process. We set 5 training epochs per user and performed 10 communication rounds between the users and the server. This resulted in a total of 50 epochs for training the local models. Each user was allocated a 10% subset of the total training dataset for training, while the remaining 10% was utilized for model evaluation.

**Table 5.** The classifier parameter values used in our proposed approach

Model	Hyperparameter	Value
DNN	Hidden dimension	128
	Output dimension	2
	Local epochs	10
	Dropout	0.2
	Convolutional layers	2 Conv1D
CNN	Kernel size	3
	Activation function	ReLu
	Local Batch Size	100
	Dropout	0.2
	Local Batch Size	100
LSTM	Activation Function	ReLu
	Hidden dimension	128
	Output dimension	2
	Dropout	0.2
	Number of Users	9
Global	Users selected	3
	Number of rounds	10
	Global epochs	5
	Global Batch size	1000
	Optimization	ADAM
	Learning rate	1e-3
	Loss function	binary cross-entropy



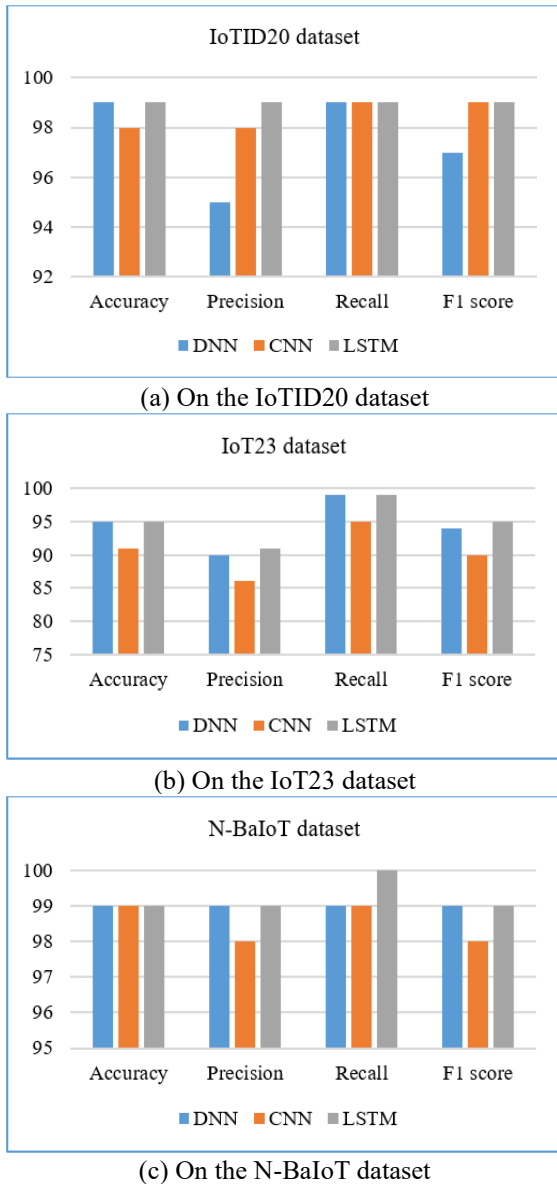
**Figure 7.** Accuracy trends of DNN training rounds on the N-BaIoT dataset

The deep learning models exhibited excellent performance during training, achieving high accuracy levels of up to 99% at the conclusion of the communication cycles. This trend is depicted in Figure 8 for all the deep learning models utilized. It is worth noting that parallel learning through federated

learning methodology enables efficient utilization of computational resources on end devices, thereby reducing overall training time while ensuring data privacy.

Following the training phase, we proceeded to evaluate the global model using the test dataset over 10 rounds. As

previously mentioned, we employed three deep learning algorithms: DNN, CNN, and LSTM. The performance of these models on the three datasets is compared in Figure 8.



**Figure 8.** Performance comparison between the DNN, CNN, and CNN federated models

We can see that on the IoTID20 dataset, as demonstrated in subgraph (a) of Figure 8, the LSTM model demonstrated the highest performance across all metrics. It achieved the highest accuracy, precision, recall, and F1-score among the three models. The DNN model performed slightly worse than the LSTM model but still showed strong results, with high accuracy and recall values. The CNN model had the lowest accuracy but still exhibited good precision, recall, and F1-score values.

On the IoT23 dataset, as demonstrated in subgraph (b) of Figure 8, the LSTM model outperformed the DNN and CNN models in terms of accuracy, precision, recall, and F1-score. It achieved the highest accuracy and F1-score among the three models, while also maintaining high precision and recall values. The DNN model performed slightly worse than the LSTM model but still achieved good results. The CNN model had the lowest accuracy, precision, and F1-score, indicating

that it may not be as effective as the other models for this particular dataset.

Finally, on the N-BaIoT datasets, as demonstrated in subgraph (c) of Figure 8, the LSTM model still performed well compared to the DNN and CNN models. It achieved high accuracy, precision, recall, and F1-score values, with the highest recall and F1-score among the three models. The DNN model achieved slightly higher accuracy, precision, recall, and F1-score than the LSTM model, indicating strong performance. The CNN model performed slightly worse than the LSTM and DNN models but still exhibited high accuracy and precision.

Overall, the LSTM model consistently demonstrated strong performance across all three datasets, outperforming or closely competing with the other models in terms of accuracy, precision, recall, and F1-score. The DNN and CNN models also exhibited good performance but generally fell behind the LSTM model in various metrics.

Finally, it is important to note that IoT data often involves time series information, and LSTM is well suited to handle such sequential data, making it an ideal candidate for intrusion detection tasks in IoT environments. The Ability of LSTM to model and remember long-term dependencies in time series data could have allowed it to identify subtle patterns and anomalies in the IoT dataset more effectively than DNN and CNN, which may have struggle to capture these temporal nuances.

#### 5.4 Comparison and discussion

Table 6 presents a comparison between the effectiveness of our approach and other IDS approaches. The comparison focuses on various aspects, including datasets, techniques employed, ML classifiers utilized, and the results obtained.

**Table 6.** Comparison of proposed model and related works

Ref.	Learning Type	Dataset Used	Classifiers	Accuracy
[39]	Centralized	CICIDS2017	CNN, LSTM	up to 97%
[40]	Centralized	CIC-IDS2017, CSE-CICIDS2018	CNN	up to 99%
[41]	Distributed	CICIDS2017	CNN, LSTM	up to 97%
[42]	Distributed	UNSW-NB15	MLP	up to 96%
[43]	Federated	NSL-KDD, DS2OS, Gas Pipeline	KNN, RF, MLP	up to 94%
[44]	Federated	MODBUS	GRU	up to 90%
Our Work	Federated	IoTID20, IoT23, N-BaIoT	DNN, CNN, LSTM	up to 99%

Our proposed FL-based SID approach for IoT networks offers several advantages compared to other approaches:

Firstly, by adopting federated learning, our approach ensures that the training process takes place locally on the user devices, preserving data privacy and security. Unlike centralized approaches where all data is sent to a central server, federated learning allows users to keep their data locally while still contributing to the model training.

Secondly, our approach leverages the power of deep learning models such as DNN, CNN, and LSTM, which have shown great success in various machine learning tasks. These models have the capability to capture complex patterns and relationships in the data, leading to improved accuracy in intrusion detection.

Furthermore, the use of FL enables distributed model training and aggregation, which reduces the communication overhead between users and the server. Instead of sending raw data, users only transmit their update parameters, reducing the risk of sensitive information being exposed or intercepted.

In terms of performance, our experiments have shown that the LSTM model, in particular, outperformed the DNN and CNN models, achieving a higher accuracy rate of up to 99% on the evaluated datasets.

Compared to traditional centralized intrusion detection approaches, our FL-based SID approach offers improved data privacy, enhanced scalability, and the ability to train robust models using distributed resources. These advantages make our approach well-suited for IoT environments where data privacy and resource constraints are critical considerations.

## 6. CONCLUSION

This study proposed a Federated Learning-based Intrusion Detection (FL-based SID) approach for IoT networks. The aim was to address the challenges of data privacy, resource constraints, and scalability in intrusion detection systems for IoT environments.

The proposed approach leveraged the power of deep learning models, including DNN, CNN, and LSTM, to capture complex patterns and relationships in IoT network data. By adopting federated learning, the training process was decentralized, allowing users to keep their data locally while contributing to the model training.

Experimental evaluations were conducted using popular IoT datasets, namely IoTID20, IoT-23, and N-BaloT. The results demonstrated the effectiveness of the FL-based SID approach, with the LSTM model outperforming the DNN and CNN models, achieving a significant accuracy rate of up to 99%.

The advantages of the FL-based SID approach included data privacy preservation, reduced communication overhead, and improved scalability. By distributing the model training and aggregation process, the approach mitigated the risks associated with sending sensitive data to a central server, while utilizing the computing resources available on IoT devices.

Finally, the FL-based SID approach presented in this study contributes to the development of efficient and privacy-preserving intrusion detection systems for IoT networks, addressing the unique challenges posed by these environments.

In future work, we intend to explore the use of other deep learning models in the FL-based SID framework. Additionally, we are considering the deployment of a federated learning architecture integrated with Blockchain technology to enhance the security and transparency of the system.

## REFERENCES

[1] Weiser, M. (1991). The Computer for the 21st Century. *Scientific American*, 265(3): 94-105.  
 [2] De, S., Barnaghi, P.M., Bauer, M.P., Meissner, S. (2011).

Service modelling for the Internet of Things. In 2011 Federated Conference on Computer Science and Information Systems (FedCSIS), Szczecin, Poland, pp. 949-955.  
 [3] Fenanir, S., Semchedine, F., Harous, S., Baadache, A. (2020). A semi-supervised deep auto-encoder based intrusion detection for IoT. *Ingénierie Des Systèmes d'Information*, 25(5): 569-577. <https://doi.org/10.18280/isi.250503>  
 [4] Vermesan, O., Friess, P., Guillemin, P., Gusmeroli, S., Sundmaeker, H., Bassi, A., Jubert, I.S., Mazura, M., Harrison, M., Eisenhauer, M., Doody, P. (2022). Internet of Things strategic research roadmap. *Internet of Things - Global Technological and Societal Trends from Smart Environments and Spaces to Green Ict*. <https://doi.org/10.1201/9781003338604-2>  
 [5] Fenanir, S., Semchedine, F., Baadache, A. (2019). A machine learning-based lightweight intrusion detection system for the Internet of Things. *Revue d'Intelligence Artificielle*, 33(3): 203-211. <https://doi.org/10.18280/ria.330306>  
 [6] Bertino, E., Choo, K.K.R., Georgakopolous, D., Nepal, S. (2016). Internet of Things (IoT) smart and secure service delivery. *ACM Transactions on Internet Technology (TOIT)*, 16(4): 22. <https://doi.org/10.1145/3013520>  
 [7] Mitchell, R., Chen, I.R. (2014). A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys*, 46(4): 55. <https://doi.org/10.1145/2542049>  
 [8] Ioulianou, P., Vasilakis, V., Moscholios, I., Logothetis, M. (2018). A signature-based intrusion detection system for the Internet of Things. *Information and Communication Technology Form*, 11-13. <https://eprints.whiterose.ac.uk/133312/>.  
 [9] Damopoulos, D., Menesidou, S.A., Kambourakis, G., Papadaki, M., Clarke, N., Gritzalis, S. (2011). Evaluation of anomaly-based IDS for mobile devices using machine learning classifiers. *Security and Communication Networks*, 5(1): 3-14. <https://doi.org/10.1002/sec.341>  
 [10] Attota, D.C., Mothukuri, V., Parizi, R.M., Pouriyeh, S. (2021). An ensemble multi-view federated learning intrusion detection for IoT. *IEEE Access*, 9: 117734-117745. <https://doi.org/10.1109/ACCESS.2021.3107337>  
 [11] Aledhari, M., Razzak, R., Parizi, R.M., Saeed, F. (2020). Federated learning: a survey on enabling technologies, protocols, and applications. *IEEE Access*, 8: 140699-140725. <https://doi.org/10.1109/ACCESS.2020.3013541>  
 [12] Khan, L.U., Pandey, S.R., Tran, N.H., Saad, W., Han, Z., Nguyen, M.N.H., Hong, C.S. (2020). Federated learning for edge networks: Resource optimization and incentive mechanism. *IEEE Communications Magazine*, 58(10): 88-93. <https://doi.org/10.1109/MCOM.001.1900649>.  
 [13] Moustafa, N., Turnbull, B., Choo, K.K.R. (2018). An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet of Things Journal*, 6(3): 4815-4830. <https://doi.org/10.1109/JIOT.2018.2871719>  
 [14] Verma, A., Ranga, V. (2019). Machine learning based intrusion detection systems for IoT applications. *Wireless Personal Communications*, 111(4): 2287-2310.

- <https://doi.org/10.1007/s11277-019-06986-8>
- [15] Kumar, P., Gupta, G.P., Tripathi, R. (2020). A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks. *Journal of Ambient Intelligence and Humanized Computing*, 12(10): 9555-9572. <https://doi.org/10.1007/s12652-020-02696-3>
- [16] Pajouh, H.H., Javidan, R., Khayami, R., Dehghantanha, A., Choo, K.K.R. (2019). A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks. *IEEE Transactions on Emerging Topics in Computing*, 7(2): 314-323. <https://doi.org/10.1109/tetc.2016.2633228>
- [17] Alruhaily, N.M., Ibrahim, D.M. (2021). A multi-layer machine learning-based intrusion detection system for wireless sensor networks. *International Journal of Advanced Computer Science and Applications*, 12(4): 281-288. <https://doi.org/10.14569/ijacsa.2021.0120437>
- [18] Almogren, A.S. (2020). Intrusion detection in Edge-of-Things computing. *Journal of Parallel and Distributed Computing*, 137: 259-265. <https://doi.org/10.1016/j.jpdc.2019.12.008>
- [19] Li, Y., Xu, Y., Liu, Z., Hou, H., Zheng, Y., Xin, Y., Zhao, Y., Cui, L. (2020). Robust detection for network intrusion of industrial IoT based on multi-CNN fusion. *Measurement*, 154: 107450. <https://doi.org/10.1016/j.measurement.2019.107450>
- [20] Li, W.J., Meng, W.Z., Au, M.H. (2020). Enhancing collaborative intrusion detection via disagreement-based semi-supervised learning in IoT environments. *Journal of Network and Computer Applications*, 161: 102631. <https://doi.org/10.1016/j.jnca.2020.102631>
- [21] Abdalgawad, N., Sajun, A., Kaddoura, Y., Zualkernan, I.A., Aloul, F. (2022). Generative deep learning to detect cyberattacks for the IoT-23 dataset. *IEEE Access*, 10: 6430-6441. <https://doi.org/10.1109/access.2021.3140015>
- [22] Campos, E.M., Saura, P.F., González-Vidal, A., Hernández-Ramos, J.L., Bernabé, J.B., Baldini, G., Skarmeta, A. (2022). Evaluating federated learning for intrusion detection in Internet of Things: Review and CHALLENGES. *Computer Networks*, 203: 108661. <https://doi.org/10.1016/j.comnet.2021.108661>
- [23] Kumar, P., Silambarasan, K. (2019). Enhancing the performance of healthcare service in IoT and cloud using optimized techniques. *IETE Journal of Research*, 68(2): 1475-1484. <https://doi.org/10.1080/03772063.2019.1654934>
- [24] Gadasin, D.V., Koltsova, A.V., Gadasin, D.D. (2021). Algorithm for building a cluster for implementing the “memory as a service” service in the IoT concept. In *2021 Systems of Signals Generating and Processing in the Field of on Board Communications*, Moscow, Russia, pp. 1-6. <https://doi.org/10.1109/ieeeconf51389.2021.9416112>
- [25] Li, W., Shen, G., Zhang, J., Liu, D., Choi, C. (2021). A LoRaWAN monitoring system for large buildings based on embedded edge computing in indoor environment. *Concurrency and Computation: Practice and Experience*, 35(16): e6306. <https://doi.org/10.1002/cpe.6306>
- [26] Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5): 637-646. <https://doi.org/10.1109/JIOT.2016.2579198>
- [27] Dastjerdi, A.V., Gupta, H., Calheiros, R.N., Ghosh, S.K., Buyya, R. (2016). Fog computing: Principles, architectures, and applications. *Internet of Things*, 61-75. <https://doi.org/10.1016/b978-0-12-805395-9.00004-6>
- [28] Prabavathy, S., Sundarakantham, K., Shalinie, S.M. (2018). Design of cognitive fog computing for intrusion detection in Internet of Things. *Journal of Communications and Networks*, 20(3): 291-298. <https://doi.org/10.1109/jcn.2018.000041>
- [29] Atlam, H.F., Walters, R.J., Wills, G.B. (2018). Fog computing and the internet of things: A review. *Big Data and Cognitive Computing*, 2(2): 10. <https://doi.org/10.3390/bdcc2020010>
- [30] AbdulRahman, S., Tout, H., Ould-Slimane, H., Mourad, A., Talhi, C., Guizani, M. (2020). A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal*, 8(7): 5476-5497. <https://doi.org/10.1109/JIOT.2020.3030072>
- [31] Drainakis, G., Katsaros, K.V., Pantazopoulos, P., Sourlas, V., Amditis, A. (2020). Federated vs. centralized machine learning under privacy-elastic users: A comparative analysis. *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, USA, pp. 1-8. <https://doi.org/10.1109/nca51143.2020.9306745>
- [32] Asad, M., Moustafa, A., Ito, T. (2021). Federated learning versus classical machine learning: A convergence comparison. *arXiv preprint arXiv:2107.10976*. <https://doi.org/10.48550/arXiv.2107.10976>
- [33] Mothukuri, V., Parizi, R.M., Pouriye, S., Huang, Y., Dehghantanha, A., Srivastava, G. (2021). A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115: 619-640. <https://doi.org/10.1016/j.future.2020.10.007>
- [34] McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A. (2017). Communication-efficient learning of deep networks from decentralized data. *PMLR*, 54: 1273-1282.
- [35] Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V. (2020). Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2: 429-450.
- [36] Ullah, I., Mahmoud, Q.H. (2020). A scheme for generating a dataset for anomalous activity detection in IoT networks. In *33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada*, pp. 508-520. [https://doi.org/10.1007/978-3-030-47358-7\\_52](https://doi.org/10.1007/978-3-030-47358-7_52)
- [37] Garcia, S., Parmisano, A., Erquiaga, M.J. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic. *Zenodo*. <https://doi.org/10.5281/zenodo.4743746>
- [38] Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., Elovici, Y. (2018). N-BaIoT-network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3): 12-22. <https://doi.org/10.1109/MPRV.2018.03367731>
- [39] Elnakib, O., Shaaban, E., Mahmoud, M., Emara, K. (2023). EIDM: Deep learning model for IoT intrusion detection systems. *The Journal of Supercomputing*, 79: 13241-13261. <https://doi.org/10.1007/s11227-023-05197-0>

- [40] Okey, O.D., Melgarejo, D.C., Saadi, M., Rosa, R.L., Kleinschmidt, J.H., Rodriguez, D.Z. (2023). Transfer learning approach to IDS on cloud IoT devices using optimized CNN. *IEEE Access*, 11: 1023-1038. <https://doi.org/10.1109/access.2022.3233775>
- [41] Roopak, M., Tian, G., Chambers, J. (2019). Deep learning models for cyber security in IoT networks. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA, pp. 452-457. <https://doi.org/10.1109/CCWC.2019.8666588>
- [42] Hoang, T.M., Thi, T.L.L., Quy, N.M. (2023). A novel distributed machine learning model to detect attacks on edge computing network. *Journal of Advances in Information Technology*, 14(1): 153-159. <https://doi.org/10.12720/jait.14.1.153-159>
- [43] Chatterjee, S., Hanawal, M.K. (2022). Federated learning for intrusion detection in IoT security: A hybrid ensemble approach. *International Journal of Internet of Things and Cyber-Assurance*, 2(1): 62-86. <https://doi.org/10.1504/IJITCA.2022.124372>
- [44] Mothukuri, V., Khare, P., Parizi, R.M., Pouriyeh, S., Dehghantanha, A., Srivastava, G. (2021). Federated-learning-based anomaly detection for IoT security attacks. *IEEE Internet of Things Journal*, 9(4): 2545-2554. <https://doi.org/10.1109/JIOT.2021.3077803>