Islamic University of Technology (IUT)

Department of Computer Science and Engineering (CSE)

Attack and Anomaly Detection in IoT Devices using Federated Learning

Authors

Mosabbir Sadman Adib, 180042106

Moshiur Rafi, 180042114

MD Jabear Hossain Pranto, 180042140

Supervisor

Dr. Md. Moniruzzaman

Assistant Professor, Department of CSE,

Islamic University of Technology (IUT)

Imtiaj Ahmed Chowdhury

Lecturer

Department of Computer Science and Engineering

Islamic University of Technology (IUT)

A thesis submitted to the Department of CSE

in partial fulfillment of the requirements for the degree of B.Sc.

in SWE

Academic Year: 2021-2022

20th May, 2023

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by *Mosabbir Sadman*, *Moshiur Rafi* and *Jabear Hossain* under the supervision of *Dr.Md.Moniruzzaman*, Assistant Professor of the Department of Computer Science and Engineering (CSE) and *Imtiaj Ahmed Chowdhury*, Lecturer of the Department of Computer Science and Engineering (CSE) Islamic University of Technology (IUT), Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Mosabbir Sadman Adib

Student ID: 180042106

Moshiur Rafi

Student ID: 180042114

MD Jabear Hossain Pranto

Student ID: 180042140

Supervisor:

____

Dr. Md. Moniruzzaman

Assistant Professor

Department of Computer Science and Engineering

Islamic University of Technology

____

Imtiaj Ahmed Chowdhury

Lecturer

Department of Computer Science and Engineering

Islamic University of Technology

Acknowledgement

We would like express our gratitude towards IUT authority for granting us the fund and providing assistance required to implement our proposed system. We are indebted to our supervisors, Dr. Md. Moniruzzaman and Imtiaj Ahmed Chowdhury for providing us with insightful knowledge and guiding us at every stage of our journey, Finally, we would like to express our heartiest appreciation towards our family members for their continuous support, motivation, suggestions and help, without which we could not have achieved this progress in our work.

Abstract

There has been a lot of focus from governments, universities, and businesses in recent years on the intersection of cybersecurity and machine learning (ML) for the Internet of Things (IoT). The Internet of Things (IoT) can be made more secure and efficient in the future through the groundbreaking concept of federated cybersecurity (FC). This new idea has the ability to efficiently identify security problems, implement countermeasures, and contain them within the IoT network infrastructure. Cybersecurity goals are met through the federation of a shared and learned model among several actors. Protecting the insecure IoT environment requires privacy-aware ML models like federated learning (FL).

Contents

List of Figures

1 INTRODUCTION

1.1 What is IoT?

The use of IoT devices is expanding quickly day by day. The term "Internet of Things" (IoT) refers to the concept of connecting common things and devices to the Internet so they can share and gather data. It entails giving physical things intelligence and the ability to interact with their surroundings by integrating sensors, motors, resources, and network connectivity into them. IoT's main goal is to make it possible for devices to interact and work together with people and other objects digitally. This connectivity enables a seamless connection between the physical and digital worlds, resulting in a network of systems and devices that can be monitored, managed, and optimized from a distance.

1.2 Current LandScape of IoT

The Internet of Things (IoT) is rapidly developing, and with it comes an increased risk of cyber attacks. IoT devices are interconnected, and any device can be used by fraudsters to gain access to large networks. According to a Comcast study, the threat landscape is developing faster than customer protection solutions. As witnessed in previous instances, the interconnectedness and widespread use of IoT devices increases the impact of cyber attacks. As a result, anticipating and preventing IoT-related cyber attacks is critical. BY 2024, it is estimated that the current size of will be worth over 400 billion dollars.

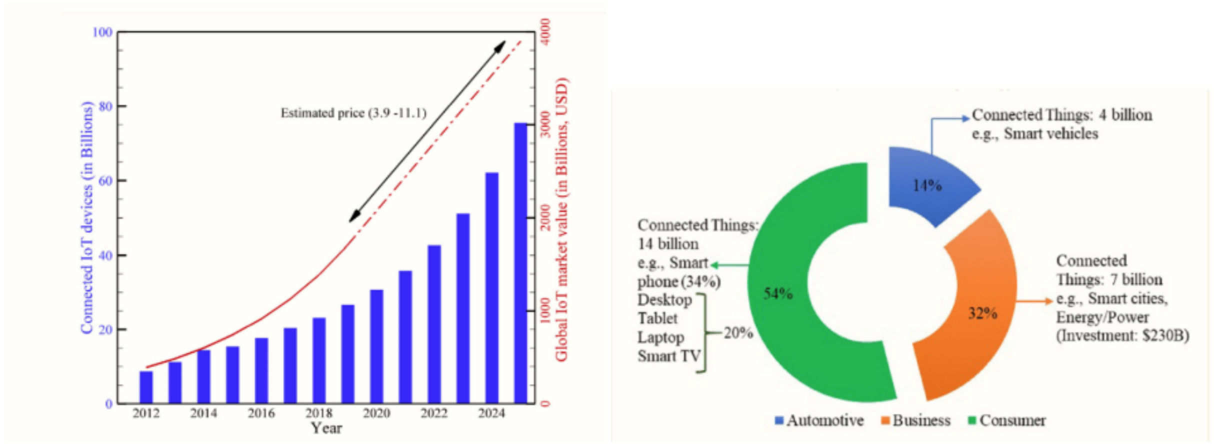


Figure 1: Growth and Market Value of IoT Devices

1.3 Concerns Regarding IoT

However, as IoT devices grow rapidly, more data is being produced every few seconds. The enormous amount of data has made it easier to use machine learning (ML) models generally in several IoT applications. Devices may gather and analyze data, make predictions, and modify their behavior in response to observable patterns and trends by combining ML algorithms with IoT systems. IoT devices may become more intelligent, more autonomous, and able to adapt to changing situations thanks to the incorporation of machine learning. It enables these devices to undertake decisions based on data, increase productivity, and provide users with individualized experiences.

Machine Learning techniques have become widely used in recent eras because of three major factors: the availability of Big Data, huge improvement in computational power, and the emergence of deep learning models. Despite ML's impressive track record, many industries can't yet leverage its capabilities in real time because of the following bottlenecks.

1. Data privacy concerns;

2. Since the data should be collected by the server for centralised learning, user privacy is jeopardised;

In machine learning, the data has to be sent to the central server for computation. But most IoT devices contain very personalized user-specific data. These can be user health or finance-related data which are very private and sensitive information. If this information leaks or steal, it can hamper the life of many users to some extent.

To overcome the above challenges, federated learning (FL) has been proposed as a new ML concept. Conceptually, FL enables different devices to learn a collaborative ML model without the need for data sharing with the centralized server.[3] [4] [5]. IoT devices can collectively learn from one another's experiences without disclosing sensitive information by using federated learning for attack and anomaly detection. With this strategy, IoT networks' security and privacy are improved, and threats and abnormalities can be effectively detected and mitigated.

1.4 Federated Learning

Federated Learning (also known as collaborative learning) is a privacy preserving machine learning technique that :

- Performs training on an algorithm across a network of edge devices that each contain local data samples but do not share.
- Enables multiple actors to build a common, robust machine learning model without sharing data

- Allows to address critical issues such as data privacy, data security, data access rights and access to heterogeneous data. [6]

2 Literature Review

2.1 Federated Learning-based Anomaly Detection for IoT Security Attacks

Using a decentralised federated learning approach with an ensembler, the authors of this paper propose a novel and inventive method for anomaly detection on IoT networks. This method is especially advantageous because it enables the training of the machine learning (ML) model on the device itself, rather than necessitating the transfer of data to a central server. This not only reduces the quantity of data that must be transmitted over the network, but also protects the data's confidentiality and security. The authors intend to train the model using advanced neural network models such as Long Short Term Memory (LSTMs) and Gated Recurrent Units (GRUs). These models are ideally adapted for sequence data, which occurs frequently in IoT networks. Extensive experiments have demonstrated that this method has a lower error rate for predicting attacks and fewer false alarms than a conventional centralised ML approach. This is an encouraging development for the field of IoT security, as it demonstrates the potential for decentralised machine learning to enhance the precision and efficiency of anomaly detection in these networks. [7] [8]

The proposed architecture for this study, as shown in Fig. 2, is composed of several key components. Firstly, the architecture utilizes virtual IoT instances that represent the devices in the network. Each of these virtual instances is equipped with a local deep learning model, which is trained on a local copy of the training data. This is a key aspect of the decentralized federated learning approach that the authors have proposed.[7]

The local deep learning models are then used to generate predictions, which are sent

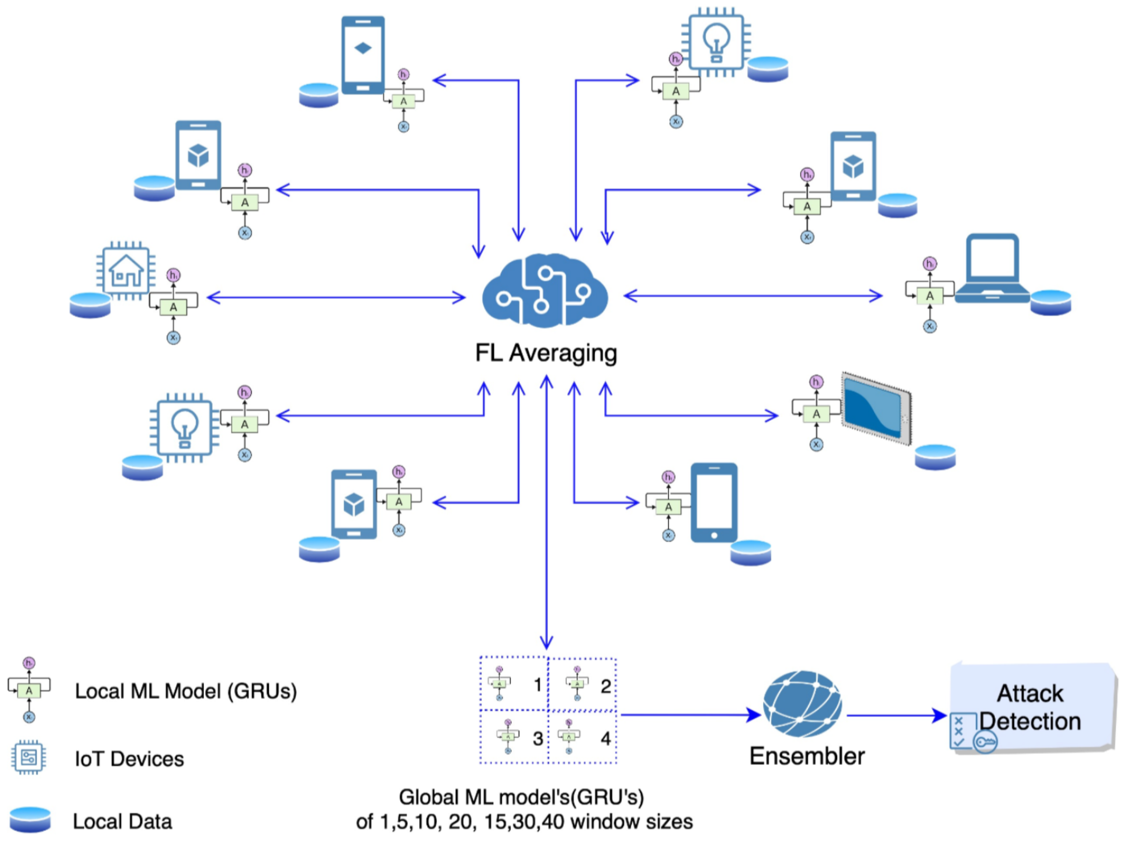


Figure 2: High-Level Architecture of the proposed approach

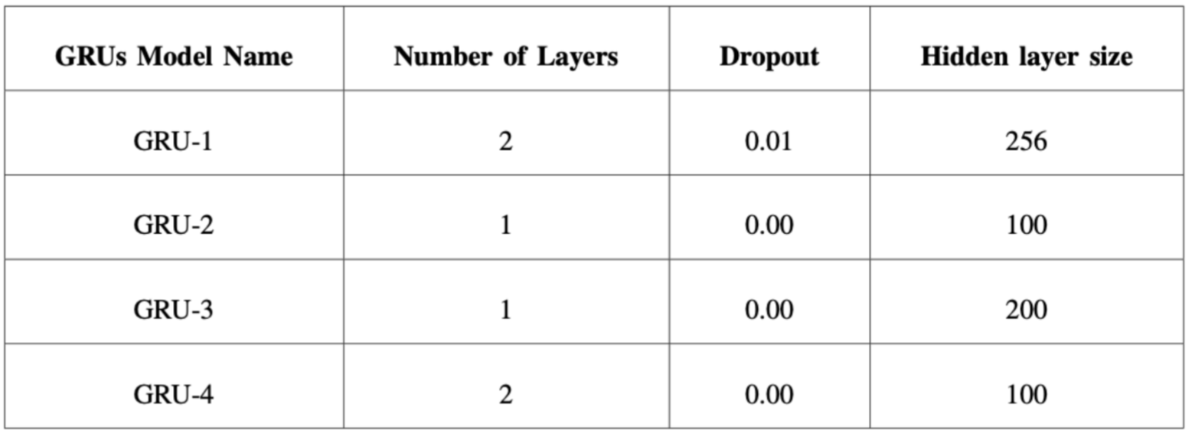
to a central server. At the central server, there is an averaging component that is used for federated learning. This component takes the predictions from all of the local models and averages them together to create a global model. The global model is then used to make predictions for the entire network.[9]

A global deep learning model is in the architecture. This allows the system to adapt to different temporal patterns in the data, which is important for anomaly detection in IoT networks. This component is used to combine the predictions from the various models and make a final decision about whether an anomaly has occurred. Overall, this architecture offers a comprehensive and robust approach to anomaly detection in IoT networks, utilizing both deep learning models and ensemble techniques to achieve high accuracy and low false alarm rates.[7]

They set up an IoT network by creating virtual instances using PySyft. They create n virtual instances for n end devices, and a dedicated instance called flaverage to simulate a central server. The dataset is divided into n chunks and distributed to the virtual instances, to enable sharing of trained ML model parameters between the end devices and the central FL server.

Using a programme called CI-CFlowmeter, the authors capture network data in pcap files and transform it into CSV files for further analysis..[8] Then the processed data is divided into n chunks and distributed among the virtual instances of the IoT end-devices.[8]

The Federated Learning training is conducted simultaneously with the available IoT instances. Each client node performs training rounds using their copy of the dataset and shares the weights of the trained local ML model with the flaverage instance for aggregation. In this study, the authors used 4-GRUs with layer size and hidden size as specified in Table II. [8]



GRUs Model Name	Number of Layers	Dropout	Hidden layer size
GRU-1	2	0.01	256
GRU-2	1	0.00	100
GRU-3	1	0.00	200
GRU-4	2	0.00	100

Figure 3: Table II: GRUs

Algorithm 1 represents the formal steps for their proposed approach and the below lists the brief steps of the FL training logic.

1. Define the window size Wi.
2. Virtual Instance created.
3. The GRU network ML model parameters (GRUML) for each window size Wi is defined
4. Share GRUML with each virtual instance fli.
5. The global ML model for each window is obtained
6. Share a duplicate of the Global ML models to each endpoint. Each virtual instance executes training rounds on a distinct processor and periodically shares learned local model weights mwi with flaverage. The training rounds have been implemented on a multiprocessor.

The authors of the paper use Ensemble Learning to attain a higher rate of accuracy by combining the output of multiple ML models. Seven global ML models Mwi are combined using a Random Forest decision tree classifier (rfc8). The models predict the probabilities of each label Y given input X for network data inputs. The Ensembler combines these probability values to create a prediction function f(x) that uses predictions as a vote for the labels of each model. The prediction function f(x) of the Random Forest decision tree classifier (rfc) uses the prediction probability values of the seven ML models for every attack category in the dataset as input. The RFC then evaluates each probability as a vote from each ML model and

```
Input: ML Local models of GRUs
Output: Network flow anomaly detection
$W = w_1, w_5, w_{10}, w_{15}, w_{20}, w_{30}, w_{40}$   /* window sizes */
$FL = fl_1, fl_2 ... fl_n$   /* Virtual IoT devices */
$m_{w_i} = m_{w1}, m_{w5}, m_{w10}, m_{w15}, m_{w20}, m_{w30}, m_{w40}$   /* local model weights for each window size */
$M_{w_i} = M_{w1}, M_{w5}, M_{w10}, M_{w15}, M_{w20}, M_{w30}, M_{w40}$   /* Global ML model weights for each window size */
Function $FL_{Training}$ ($maxtrainingrounds$):
    while $fl_i$ in $FL$ do
        foreach $w_i$ in $W$ do
            $m_{w_i}$ = train($fl_i$ in data($w_i$))   /* Train LSTM with local data */
    $return\ m_{w_i}$
    $EndFunction$
Function $fl_{average}$ ($m_{w_i}$):
    foreach $w_i$ in $W$ do
        $M_{w_i} = fl_{average}(m_{w_i})$
    $return\ M_{w_i}$
    $EndFunction$
Function $Ensembler$ ($M_{w_i}$):
    $networkdata$   /* New flows in-network data */
    foreach $w_i$ in $W_i$ do
        $lstmpredictions = m_{w_i}(newnetworkdata)$
        $anomalydetectionflag = Ensembler(lstmPredictions)$
    $EndFunction$
$m_{w_i} = FL_{Training}(maxtrainingrounds)$
$M_{w_i} = fl_{average}(m_{w_i})$
while $fl_i$ in $FL$ do
    foreach $w_i$ in $W$ do
        $m_{w_i} = M_{w_i}$   /* replace local ML */
$Attack_{Prediction} = Ensembler(M_{w_i})$
```

Figure 4: Algorithm 1: Anomaly detection with federated learning

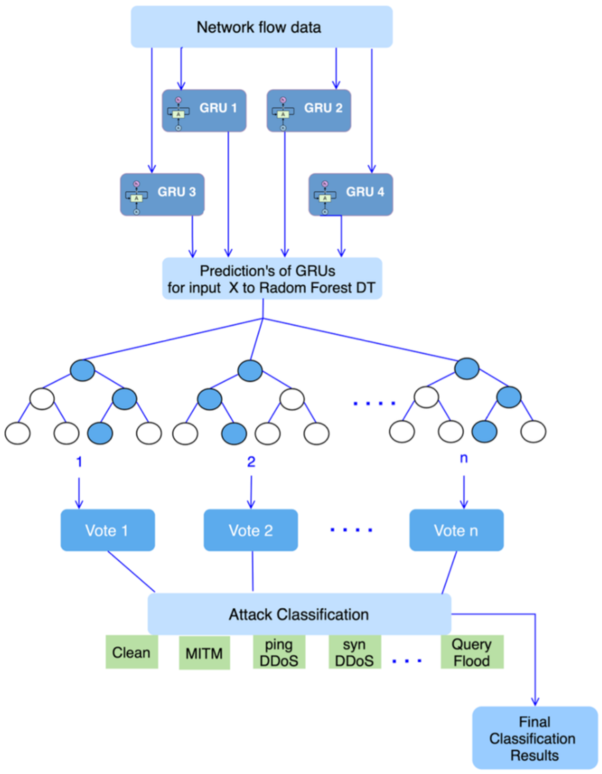


Figure 5: Illustration of Ensembler in proposed approach

predicts the output label with a high degree of confidence. The Ensembler is incorporated into the proposed method, as shown in Figure 4. [8]

The total time required to complete a training round in Federated Learning includes the extra time needed for the FL aggregation process and the time spent communicating with each FL client. The above formulas give the calculations.[8] The authors of the paper used Skorch and scikit-learn for evaluating the trained ML models. As their approach is based on the Pytorch deep learning framework, they used the skorch wrapper to access the evaluation packages provided by the scikit-learn framework. [8] [9]

Compared to a non-FL implementation, the proposed method performs better, as shown by the evaluation results. The authors have implemented FL with a custom ML model for the sake of demonstration, but in a real-world production setting, the global model can be pre-trained with examples of common attacks and a training dataset. This improves the global model's capacity to anticipate the likelihood of an attack and keeps it up-to-date. Virtual instances, which can only use data collected in advance of the training round and cannot produce live logs, were used to assess the method. The evaluation findings imply that the suggested approach can further improve attack detection categorization in a practical scenario with a productionized version of IoT networks, despite the restrictions of virtual instances and the additional time for local model averaging.[8]

2.2 Federated Learning for Resource-Constrained IoT Devices

The authors discuss the challenges and limitations of Federated Learning (FL). They also highlight that clients may lack the resources for on-device computation, leading to slow convergence.

In an Internet of Things (IoT) ecosystem built on Federated Learning (FL), communication overhead is a significant obstacle. The iterative and non-optimized way to communication between the server and the clients, combined with the enormous quantities of data passing during the process, drives up the communication cost. Clients with insufficient resources, such as low bandwidth, make it more difficult for the client to communicate with the FL server during model update, exacerbating the situation

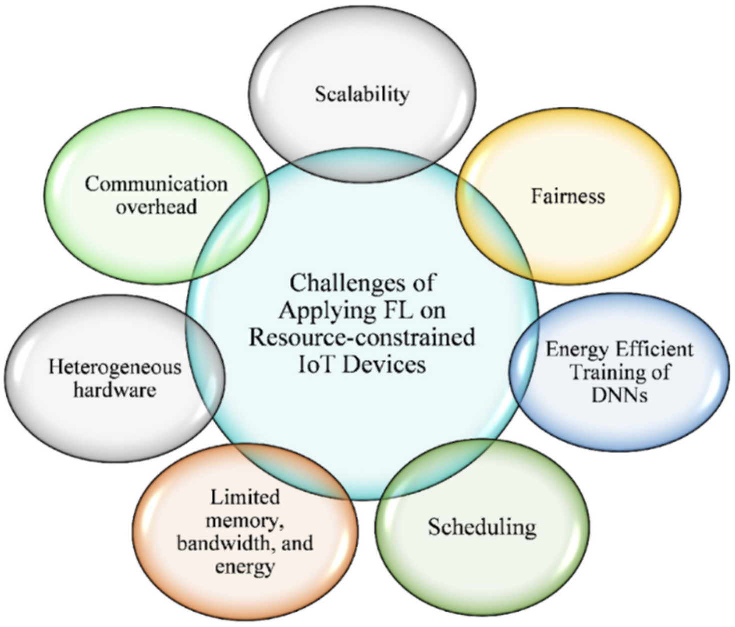

Scalability
Communication overhead
Fairness
Challenges of Applying FL on Resource-constrained IoT Devices
Heterogeneous hardware
Energy Efficient Training of DNNs
Limited memory, bandwidth, and energy
Scheduling

Figure 6: Core challenges of FL considering resource-contraint IoT devices

There are various studies that propose frameworks for fair accuracy distribution, resource allocation and incentive mechanisms in FL. These frameworks include minimax optimization framework, -fairness metric and q-Fair FL, collaborative fair FL framework, FL-based client selection process, incentive-based FL model, payoff-sharing scheme, trust and incentive-based FL model, client contribution-based incentive method. These frameworks aim to ensure fairness in FL resource allocation by considering clients' activities, resource status, and contributions toward model convergence, reduce communication overhead and computation power, and to achieve higher accuracy.[7] [10]

In Federated Learning (FL) for IoT environments, there are challenges in dealing with a large number of heterogeneous devices with limited resources. To address these challenges, researchers have proposed various strategies for effective client selection and optimal resource utilization. Some of the proposed strategies include joint learning and wireless communication, client selection protocols based on resource conditions and trust scores, selective client model aggregation, and trilayer lightweight frameworks to handle large numbers of clients and data streams. These proposed strategies aim to ensure scalability, robustness, and sparsification of the FL process. [11]

The authors address the tradeoffs and limitations of federated learning (FL) by discussing various strategies for lowering communication costs within FL. Decentralised training, model compression, and priority-based updates are all examples of such methods. The authors note that combining these tactics can help make up for this field's tradeoffs and deficiencies. They also note that successful optimisation strategies, formally formalised in theory, then implemented and tested empirically, can make the FL method more scalable. However, the authors acknowledged that most studies they cited ignored the difficulties caused by the diversity of client resources.

When the training data across devices are not identical in terms of data modelling and convergence behaviour, FL training becomes complicated. Several studies have concentrated on the design of statistical heterogeneity through meta-learning and multitask learning, which have been extended to FL settings. [7]

In FL-based IoT environment, there are several challenges that need to be addressed in future research. These challenges include dealing with discrepancy in data samples among clients, ensuring convergence for asynchronous learning in a Non-IID setting in the presence of resource-constrained devices, cluster head selection for energy efficiency and minimizing bandwidth requirements, device-centric automatic wake-up mechanism for energy conservation, handling client mobility and statistical heterogeneity among client data, and designing effective incentive mechanisms to encourage clients to share their model information. [8]

2.3 Communication-Efficient Learning of Deep Networks from Decentralized Data

Digital device data is generally highly private, vastly abundant, or both, which may prevent login to the data center and training there using traditional methods. The paper discusses a learning technique that enables users to profit from shared models created from this rich data collectively without having to store it centrally which is Federated Learning (FL). In FL, each client computes an update to the current global model kept by the server instead of sending data to the server, and only this update is sent.

This paper introduces an algorithm named FederatedAveraging algorithm or FedAvg. It combines model averaging on the server with local stochastic gradient descent (SGD) on each client. Basically here all the clients shared a global model and train the model collaboratively. [12] The paper also discusses a few optimizations that can significantly improve the performance of the system. Numerous operational concerns must also be addressed by a deployed federated optimization system, including Client datasets that fluctuate as data is added and removed; client availability that tightly ties to the local data distribution; and clients that never reply or send updates that are corrupted.

FedAvg algorithm performs aggregation and model update based on the minimization of global function loss which is the average of individual client loss. The following is the equation for calculating global loss:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w)$$

Here K is the total number of clients in a network, and Fk is the loss function of the client K. Each loss function is then weighted by the size of the client k's dataset. So larger datasets will correspondingly have larger data losses.

While communication costs are quite minimal in centralized learning, computational costs predominate. Recent years have seen a lot of focus on leveraging GPUs to cut these expenses. In contrast, communication costs rule in federated optimization; frequently, our upload bandwidth is limited to 1 MB/s or less. Additionally, customers frequently elect to participate in the optimization only when they are powered on, charged, and connected to an unmetered wifi network. As a result, the major goal is to use more computation to cut down on the number of communication rounds needed to train a model. Let's briefly discuss the implementation of the FedAvg algorithm and how it works internally

```
**Server executes:**
  initialize $w_0$
  **for** each round $t = 1, 2, \dots$ **do**
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of $m$ clients)
    **for** each client $k \in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow$ ClientUpdate$(k, w_t)$
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

**ClientUpdate**$(k, w)$**:**   // *Run on client k*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)
  **for** each local epoch $i$ from 1 to $E$ **do**
    **for** batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return $w$ to server
```

Figure 7: Federated Averaging algorithm

[13] Stochastic Gradient Descent is a general-purpose optimization approach that can solve a variety of problems at their optimum. It can be naively used to solve the federated optimization issue, where each cycle of communication involves a single batch gradient calculation. Although this method is computationally efficient, it needs a lot of training rounds to build good models.[13]

The Federated Averaging algorithm has a lot of benefits in practical life but it's not perfect. For example, it assumes all sample devices will complete all the epochs of stochastic gradient descent (SGD). But not all device has the same level of computation power to complete all the epochs within time. Some device struggles to meet the requirement. So to tackle this situation FedAvg just drops the struggles as strugglers as these hurt the speed of convergence. But what if most of our devices are strugglers?
At the same time, not all devices have the same amount of training data. So the distribution of training data is unbalanced. But FedAvg is biased towards some clients. Favors the clients with more data.

2.4 Federated Learning for Internet of Things: A Federated Learning Framework for On-device Anomaly Data Detection

In this study, a FedIoT platform containing the FedDetect algorithm was created using a straightforward but efficient design philosophy in order to assess both algorithmic and system performance in real IoT devices. The goal of the design is to ascertain whether the global model developed through FL training can distinguish between various attack types and has better detection capabilities. Instead of using the simplistic FedAvg for local training, the FedDetect algorithm employs an adaptive optimizer and a cross-round learning rate scheduler. Deep Autoencoder was used as the model for anomaly detection to evaluate FL methods and the FedIoT platform.[14]

Usually, for attack detection tasks, centralized machine learning approaches fail to provide privacy and protection of users' sensitive and confidential data. Federated learning can train a global or personalized model without centralizing edge-device data. The work's proposed platform helps researchers to perform their research in a real IoT environment. Existing works only conduct simulations as opposed to actual experimentation on IoT platforms.[14] The application layer, the algorithm layer, and the infrastructure layer make up the whole software architecture.

- Application Layer:

  FedIoT offers a one-line API to start federated training on Internet of Things devices using distributed computing.

- Algorithm Layer:

  Algorithmic benchmarks supported by FedIoT include FedAvg, FedOPT, and FedDetect.

- Infrastructure Layer:

  The FedIoT project seeks to enable simple communication APIs.

The main objective of FedDetect is to stop malware-related anomalous traffic. So it is designed to deploy on the router. Before entering the IoT devices, the security gateway will first screen the incoming traffic. It provides an extra layer of security for IoT devices.

On the N-BaIoT dataset, the FedIoT platform and FedDetect algorithm were analyzed. Nine commercial IoT devices' network traffic flow is captured by N-BaIoT. Every IoT device has two subsets: a benign set with only typical network traffic data and an attack data subset with two well-known malware threats. The guarantee of the following three data properties was the task's top priority:

- It has training information for many device types;
- Each device lacks a complete array of attack types;
- All attack types on the entire IoT network should be included in the evaluation and test dataset.

The experiment was carried out by using benign IoT traffic data to train a deep Autoencoder about the normal IoT device behavior. After training the autoencoder on the benign training dataset, reconstruction error (MSE) was calculated for each data sample from the benign evaluation dataset, and the threshold was then obtained using that.
Following the development of the global model through federated training, the Global Threshold for each device was determined using an algorithmic module. Each machine uses this approach to run the global model locally to obtain the MSE sequence, which is subsequently synchronized with the server. The Global Threshold approach provides an objective representation of the detection performance of the FL-trained global model on all IoT networks.
The evaluation FedIoT platform was done considering two aspects:

- performance of the global model's algorithms

- a thorough evaluation of the system's performance, taking into account the speed of processing, communication costs, and memory costs

First, the performance of the FedDetect algorithm was evaluated using the global threshold and compared to the three previously mentioned baselines: CL-Single, CL-Combined, and FL-Fedect. The more benign data that the detection model can train on, the better it should perform. For the purposes of the overall evaluation, the CL-Combined baseline is trained on all benign data. As a consequence, its performance is superior to all other models. FedDetect outperforms CL-Single by a wide margin, as it collects more data from all devices collectively than CL-Single. FL's FPR and TNR performance is marginally inferior to CL's and may not be appropriate for the local evaluation targets, as the direction of gradient descent is modified after aggregation and averaging during ML optimisation.

Using a globalised threshold, the second phase of the investigation assessed the FedIoT system's efficacy on the Raspberry Pi. The experiment demonstrates that the training time memory cost utilises only a minor portion of the Raspberry Pi's host memory. The segmentation of end-to-end training time was analysed to gain a deeper understanding of the system's price. Overall, it demonstrates that instruction for practical applications can be completed in less than an hour, which is a sufficient amount of time.

The results demonstrate that federated learning is effective at identifying a wide range of attack types, and an evaluation of system efficiency reveals that end-to-end training time and memory costs are reasonable and optimistic for IoT devices with constrained resources.

2.5 A Federated Learning-Based Approach for Improving Intrusion Detection in Industrial Internet of Things Networks

Problems and concerns around the security and privacy of Internet of Things (IoT) networks are addressed by the researchers in this paper. They draw attention to the IoT system's vulnerability because of the volume of data created by linked devices, which leaves it open to hostile attacks. Due to the need for enormous volumes of centralized data, the centralized machine learning (ML) methodologies employed in IoT systems threaten data privacy and are difficult to implement. The researchers suggest a Federated Learning (FL) method for identifying intrusions in IoT networks while maintaining privacy and security to get around these restrictions. In the FL method, parameter updates are shared between local IoT clients and a central global server after models are trained at the local device level. The researchers use comprehensive trials to show the dependability and efficacy of their intrusion detection model, which achieves accuracy that is nearly 92.49 percent (using the FL approach) to that of typical centralized ML models (93.92 percent). The study also highlights the difficulties the Industrial Internet of Things (IIoT) faces as well as the requirement for effective control systems that put an emphasis on energy efficiency, expanded application capabilities, and scalability. The benefits of FL in wireless IoT networks, such as decreased power consumption, decreased transmission delay, and improved data privacy, are highlighted by the researchers. The paper also clarifies how important intrusion detection systems (IDS) are for detecting and preventing attacks, as well as the security issues with IIoT devices. The suggested FL-based method provides a decentralized and effective way to improve security. [15]

The study's methodology calls for the use of federated learning (FL) to create an efficient intrusion detection system for environments leveraging the Industrial Internet of Things (IIoT). In typical contexts, training data from the objects being modeled is used to build models. The restricted functionality of IIoT devices and the difficulties in gathering enough training data make this method difficult in IIoT contexts. This problem is addressed by the suggested FL technique, which accelerates the development of a stable model by merging training data from several users into a single local training dataset.

The clients exchange their trained learning with a server-based system for aggregation throughout the learnings dissemination phase. The interactions between clients and the aggregation server are facilitated by an intelligent communication administrator, such as a security gateway. [15]

The aggregation server gathers the local learnings and converts them into global learnings under the collected global learnings section. The clients are subsequently given access to the optimized model, which facilitates information transfer and enhances the learning process. Through the use of analogous behaviors gathered from several participating devices, this sharing mechanism enables clients to detect intrusions, gradually enhancing learning.

The study makes an assumption that a reliable FL aggregator, safe clients, and no intentionally malicious IIoT devices exist. The security and integrity of the FL process are guaranteed by these presumptions.

The study uses the convolutional neural network (CNN) and the recurrent neural network (RNN) as two different types of classifiers for intrusion detection. While RNNs are used to handle data sequences and time series, CNN is used to analyze data presented as numerous arrays. Long-term memory is also incorporated into RNNs using the long short-term memory (LSTM) variation.

This study's findings demonstrate the advantages of federated learning (FL) for intrusion detection in Industrial Internet of Things (IIoT) systems. Instead of using traditional Machine Learning (ML) techniques, IIoT devices can transmit data more securely and with less bandwidth usage by using FL. FL protects user privacy and data security by spreading data so that it cannot be accessed from a single server. Additionally, independent prediction and anomaly detection are made possible even in disconnected circumstances by the local representation of models. Through collective learning from all client-end devices, the models' performance improves as the number of FL rounds increases, resulting in intrusion detection accuracy comparable to centralized ML models. The studies carried out utilizing CNN and RNN models on the Edge-IIoTset dataset show the applicability and value of the suggested FL technique. Future work will strengthen the model's defense against hostile edge nodes, add outlier detection filtering to thwart attempts at poisoning, and provide FL algorithms for resource-adequate device selection. Future research projects must also address problems including device malfunctions, sluggish upload/update times, and a lack of pertinent data if they are to increase the global model's accuracy. [15]

3 METHODOLOGY

3.1 Research Question

We implement machine learning-based attacks and anomaly detection on IoT devices on a centralized system. For a centralized machine learning-based system, we need to send users' data to the server to train the global model. However, a lot of IoT devices are heavily personalized and hold users' extremely sensitive private data. Sending this information to a server just violates user privacy and data security. So we introduce federated learning-based attacks and anomaly detection on IoT devices to solve the issue of protecting user data while ensuring the security of IoT devices.

3.2 Proposed Approach

In centralised machine learning, the client's data is often transmitted to the central server. With FL, however, the server distributes the model to the chosen clients rather than the other way around. The client then uses their own data to train the model. After the model has been trained, just its new weights and parameters will be uploaded to the server. As a result, the client's information remains largely within the client's device. After the server has received updates from all of its clients, it will use the FedAvg method to aggregate and average all of the weights. This allows us to incorporate federated learning into our processes. The server then adjusts the worldwide machine learning model based on the new information. The clients will then receive the revised model in preparation for the subsequent cycle. We won't stop until we're satisfied with the level of detail and accuracy. In the future, this worldwide model will be used to spot hacking attempts on Internet of Things gadgets.

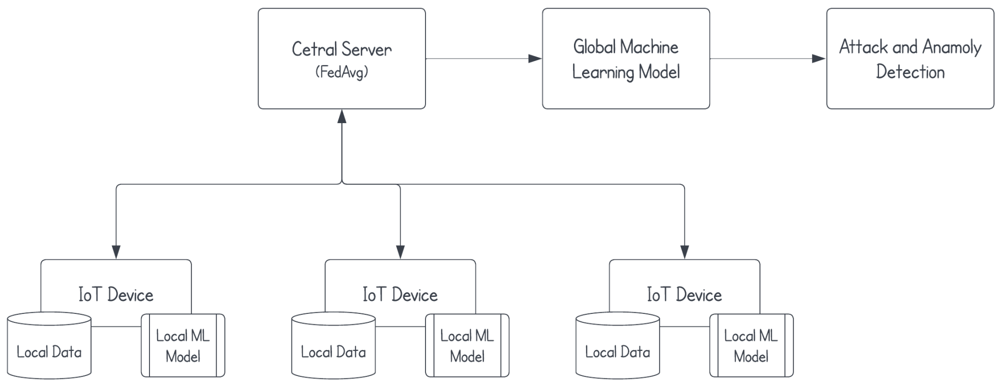


Figure 8: Proposed model workflow

3.3 Datasets

For training the model we mainly consider three very popular intrusion and attack datasets:

- NSL-KDD
  - Contains 3 925 650 attack records
  - created in 2009 to address the KDDCup99-related inefficiencies
  - Size: 15MB
- UNSW-NB15
  - Contains 2 million records
  - includes nine attack families with names like normal, fuzzes, analysis, backdoors, DoS, etc

    - Size: 156MB

- CICIDS 2017
    - Contains 2 830 540 records
    - There are 83 features per record.
    - Contains 14 attack classes.

3.4 Frameworks

We are considering using some frameworks to train our model based on federated learning. These frameworks provide us with the platform to simulate a distributed network where we can train our model on decentralized data.

- TensorFlow Federated (TFF)

  An open-source platform for decentralized data processing and machine learning

- Flower

  a coordinated framework for federated learning, analytics, and assessment. Federate any programming language, any ML framework, and any task.

3.5 Implentation

We have implemented used the TensorFlow Federated framework to implement federated learning. For the dataset, we have used the same dataset that we used in our previous work where we detect anomalies in IoT devices using a centralized server.

In our work, we have classified the data into two categories - normal data and anomaly data (this anomaly data involves data of DoS attack data, spy data, malware data, data probing data, etc). We have created two clients for the work. In federated learning, we don't share data with the server but rather share model data with the server.

So our whole working procedure can be divided into 5 steps:

- **Global Model**

  Initially, the server has an untrained model, to begin with. The server then sends the global model to each of the connected client nodes. To make sure that all of the participating nodes begin their local training with identical model parameters, this initial step is crucial.

- **Local Model Train**

  Now that each client receives the global model and the parameters we start the training process. As every client node has its own local data, they train their local model using those local models.

- **Model Trains**

  Following local training, the model parameters that each client node originally got are slightly modified. Each client node's local dataset contains a different collection of examples, which explains why the parameters are all unique. Each client now sends a copy of its trained model parameters to the server. They usually send the full model parameters or just the gradients.

- **Model Aggregation**

  Now at this step, the server receives the updates from the client's node. But as the server receives updates from a lot of clients, it has a lot of different versions of the existing global model. So here in the server aggregation is performed. **Federated Averaging (FedAvg)** is the simplest method for doing it. The model updates are averaged by **FedAvg**. The algorithm takes the weighted average of the model updates of all the clients, which is then adjusted for the number of training instances each client used. Here, weight is crucial to ensuring that no client influences the outcome of the global model.

- **Updated Global Model**

  The server has now captured the pattern of training data on all the client's nodes. The server is now sending the updated models to each node (the whole process from step-1 to step-4 will be repeated again)

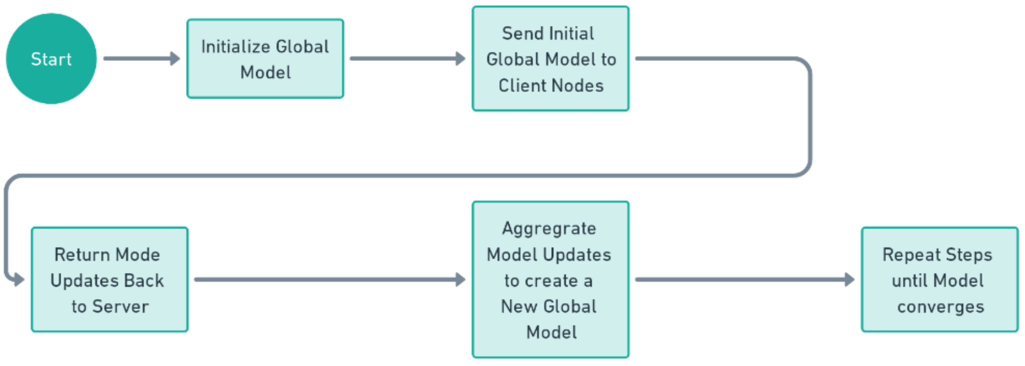

Start
Initialize Global Model
Send Initial Global Model to Client Nodes
Return Mode Updates Back to Server
Aggregrate Model Updates to create a New Global Model
Repeat Steps until Model converges

Figure 9: Our Working Procedure

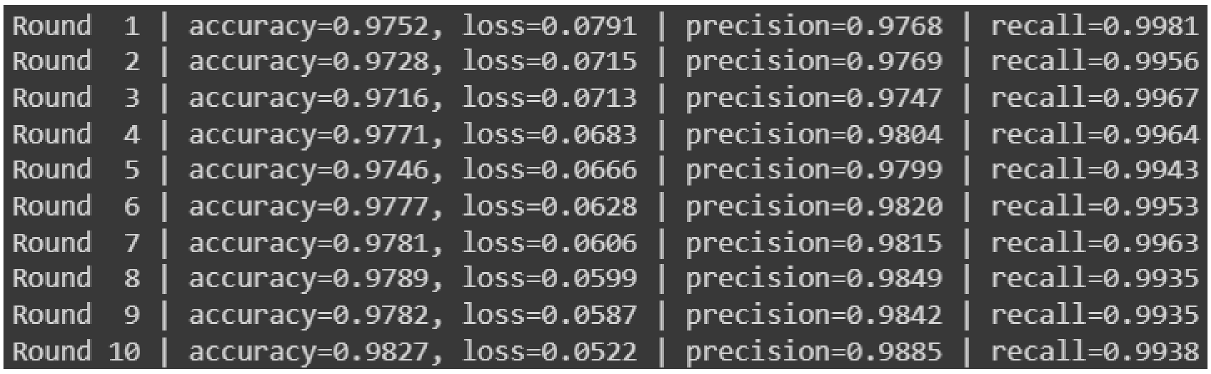
Here no data sharing is happening. Everything is achieved at the edge. No data sharing means there is privacy preservation and also very little communication overhead.

3.5.1 Working Procedure

In our research work, we first divided our data into two clients equally. Before assigning the data, we randomize the dataset to reduce biases. We preprocessed the dataset, filter out the null and missing values, and used the **SMOT** algorithm to balance the dataset. As the target column of our selected column contains string-type data, we have used **LabelEncoder** to transform those data into numerical values.

Then we eventually start the training process. We have created an 80/20 split of training and testing datasets for each client. For the model, we have used a model provided by TensorFlow that is powered by sequential neural network architecture. During the learning process, we used the **SGD** optimizer in order to update the model parameters and optimize the performance.

We then run the model for 10 rounds and got around 97% - 98% accuracy. Below is the result of our simulation



```
Round  1 | accuracy=0.9752, loss=0.0791 | precision=0.9768 | recall=0.9981
Round  2 | accuracy=0.9728, loss=0.0715 | precision=0.9769 | recall=0.9956
Round  3 | accuracy=0.9716, loss=0.0713 | precision=0.9747 | recall=0.9967
Round  4 | accuracy=0.9771, loss=0.0683 | precision=0.9804 | recall=0.9964
Round  5 | accuracy=0.9746, loss=0.0666 | precision=0.9799 | recall=0.9943
Round  6 | accuracy=0.9777, loss=0.0628 | precision=0.9820 | recall=0.9953
Round  7 | accuracy=0.9781, loss=0.0606 | precision=0.9815 | recall=0.9963
Round  8 | accuracy=0.9789, loss=0.0599 | precision=0.9849 | recall=0.9935
Round  9 | accuracy=0.9782, loss=0.0587 | precision=0.9842 | recall=0.9935
Round 10 | accuracy=0.9827, loss=0.0522 | precision=0.9885 | recall=0.9938
```

Figure 10: FL Simulation Result

3.6 Challenges

The path of federated learning is definitely not a straight one. We have to face a lot of hurdles and difficulties which are far different from what we usually see in centralized machine learning. Let's briefly discuss some of the challenges that we will possibly encounter:

- Non-IID

  The data of the selected devices in federated learning have highly heterogeneous data. The heterogeneity is from both data quality and system configuration. Therefore, no user's local dataset will accurately reflect the dispersion of the population.

- Unbalanced training data

  Some device users may use the device heavily like using a particular app frequently or taking a lots pictures than others. So the amount that both of the devices has varies a lot and hence the local training data will be unbalanced for training the model.

- Massively distributed clients

  It is expected that there would be many more clients participating than there would typically be examples per client.

- Limited Communication

  As in federated learning clients communicate with servers wirelessly, so many factors can disrupt the communication like clients are in offline mode or out of batteries. These communications are expensive and have high overheads.

The majority of these challenges are beyond the scope of our work, therefore instead, we employ a controlled setting that is appropriate for testing while still addressing the crucial problems of client accessibility and unbalanced and non-IID data.

3.7 Limitations

- One of the major flaws of our work we used some virtual instances (as client nodes) to implement federated learning. There will be definitely some complications when we going to use this model in a real environment.
- We used SMOTE algorithm to balance our dataset. There is a very limited amount of data to work with in federated learning as data is not shared in this approach. As a result, It becomes difficult to compile thorough data from numerous devices or sources in order to recognize and comprehend new attack patterns or abnormalities.
- In our research work, we are working with IoT devices that have limited communication bandwidth and computation resources. This limitation of IoT devices can hamper the training process and exchange of information on the central server. Therefore it can impact the efficiency of attack detection on those devices.

4 RESULT ANALYSIS

We took into consideration four performance indicators to evaluate the performance of our model. These are

- Accuracy

  the fraction of accurately classified instances over the total instances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision

  refers to instances that are positively classified.

$$Precision = \frac{TP}{TP + FP}$$

- Recall

  consists of actual positives that are categorized as positive instances.

$$Recall = \frac{TP}{TP + FN}$$

4.1 Centralized vs Decentralized Approach

In our research work, we have used both centralized and decentralized approaches to detect attacks and anomalies in IoT devices.

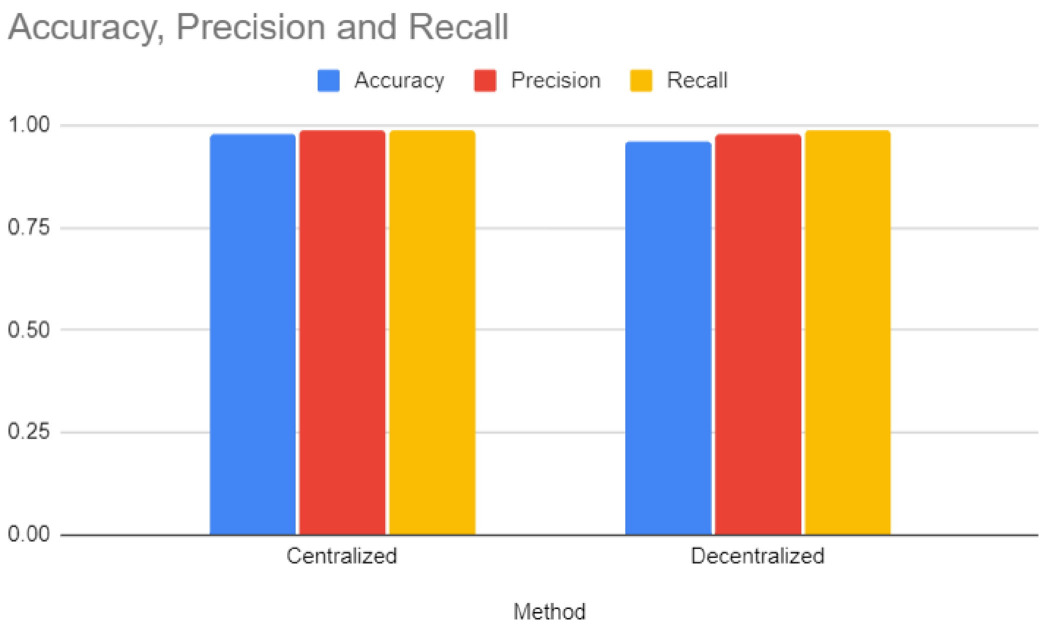


Figure 11: Centralized Vs Decentralized Approach

As we can see the difference in accuracy is pretty close in both centralized and decentralized methods. We have used the same dataset to train the model in both cases. But decentralized federated learning approach helps us in ensuring data privacy and lower communication overheads.

Our approach was evaluated using virtual instances that are unable to give live logs and instead rely only on the data provided prior to the training phase. The evaluation results show that, despite the limitations of virtual instances and add-on local model averaging time, our proposed approach can further enhance attack detection classification in the real-world scenario utilizing a productized version of IoT networks.

4.2 Our Model vs FedAnomaly Detection [1]

Here we are showing side by side comparison of our model's accuracy and this paper's [1] accuracy. They also worked on detecting attacks and anomalies on IoT devices, but our model certainly outperformed their result

Models	**Accuracy** (%)
NON-FL	86.13
FL	95.25
Our Model	**97.23**

4.3 Our Model vs FedDetect (FedIoT Platform) [2]

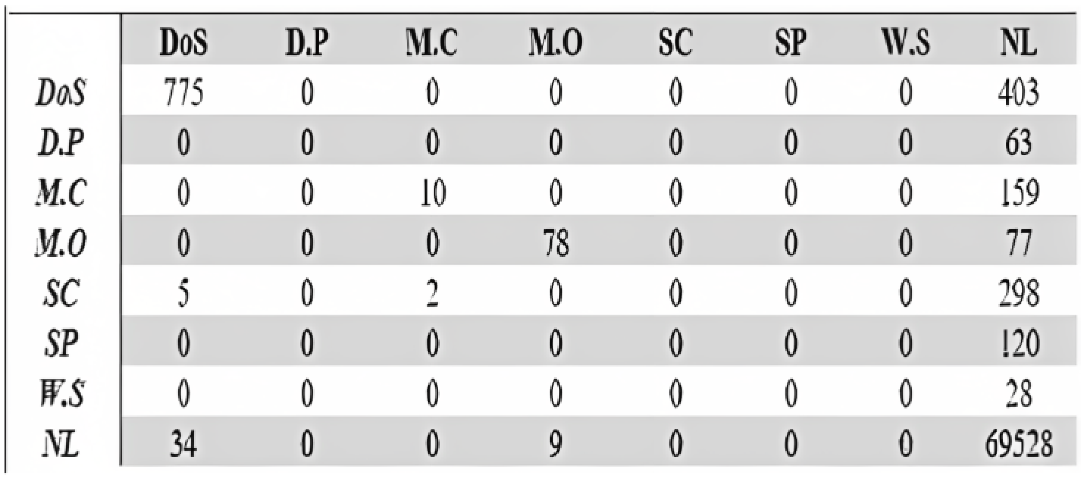
We also compare our result with this paper [2]. When the authors of the paper used the vanilla FedAvg algorithm to detect attacks and anomalies on IoT devices, their model got around 73% accurate whereas the accuracy of our model is around 97%. So there is certainly a huge improvement. But when using the FedDetect algorithm for aggregating the model updates instead of the FedAvg algorithm, the accuracy model significantly improved. FedDetect is an improved version of the FedAvg algorithm.

Models	**Accuracy** (%)
CL-Single	73.82
Fed-Detect	98.23
Our Model	97.23

But as we have used TensorFlow federated for implementing federated learning which used the FedAvg algorithm under the hood, we certainly didn't have the option to change it. But still, our model was close enough to theirs.

4.4 Attack Classification

Our dataset contains 9 classes in the attack - one normal class and 8 anomaly data classes. Based on the prediction we have created and confusion matrix which indicated the amount of data samples our model correctly predicted or how many samples are falsely classified. The elements that are on the diagonal position in the confusion matrix are correctly classified, while others are falsely classified. We have balanced the dataset to some extent, but still, there remain some classes with very little sample compared to others which were the ones that were unclassified.



	DoS	D.P	M.C	M.O	SC	SP	W.S	NL
DoS	775	0	0	0	0	0	0	403
D.P	0	0	0	0	0	0	0	63
M.C	0	0	10	0	0	0	0	159
M.O	0	0	0	78	0	0	0	77
SC	5	0	2	0	0	0	0	298
SP	0	0	0	0	0	0	0	120
W.S	0	0	0	0	0	0	0	28
NL	34	0	0	9	0	0	0	69528

Figure 12: Confusion Matrix of Federated Learning Result

5 Conclusion

Poison attacks are a type of cyber attack that involves an attacker injecting fake information or data into a system or network. The attacker aims to corrupt the system or network with the fake data, which can cause harm to users. There are different types of poison attacks, including cache poisoning and web cache poisoning. We need to work on detecting poison attacks on Federated Learning

- Work on strategies to mitigate poison attacks in Federated Learning.
- Use FedDetect Algorithm and identify attacks.

Because IoT devices continuously generate data at the edge, there is a vast amount of data produced by a variety of IoT devices that is widely accessible and transferrable to the main server. As a result, it is the least desirable option for domains with concerns about the privacy of user data because conventional ML relies on the legacy set of all data that is stored on a single server. We suggest an FL-based anomaly detection strategy to solve this problem. Our proposed work provides a solution for attack detection on IoT devices without using users' personal and sensitive information on a centralized server. The machine learning model we built will be trained on decentralized data thanks to federated learning. IoT networks get a safe layer due to the FL benefits of user data privacy, increasing the dependability of IoT devices. The federated learning-based approach has a lot of challenges and difficulties in its path. Solving all these challenges is beyond the scope of our work. Our main goal in this research is to identify a variety of attacks and threats that IoT devices potentially encounter. The performance of our model will be enhanced over time.

References

[1] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, “Federated-learning-based anomaly detection for iot security attacks,” *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2545–2554, 2021.

[2] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain, “Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach,” *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6348–6358, 2021.

[3] M. Alazab, S. P. RM, M. Parimala, P. K. R. Maddikunta, T. R. Gadekallu, and Q.-V. Pham, “Federated learning for cybersecurity: Concepts, challenges, and future directions,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3501–3509, 2021.

[4] B. Ghimire and D. B. Rawat, “Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things,” *IEEE Internet of Things Journal*, 2022.

[5] I. Siniosoglou, P. Sarigiannidis, V. Argyriou, T. Lagkas, S. K. Goudos, and M. Poveda, “Federated intrusion detection in ng-iot healthcare systems: An adversarial approach,” in *ICC 2021 - IEEE International Conference on Communications*, pp. 1–6, 2021.

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

[7] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2022.

[8] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated-learning-based anomaly detection for iot security attacks," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2545–2554, 2022.

[9] B. Alhalabi, S. Basurra, and M. M. Gaber, "Fednets: Federated learning on edge devices using ensembles of pruned deep neural networks," *IEEE Access*, vol. 11, pp. 30726–30738, 2023.

[10] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "Iot security techniques based on machine learning: How do iot devices use ai to enhance security?," *IEEE Signal Processing Magazine*, vol. 35, no. 5, pp. 41–49, 2018.

[11] R. Ito, M. Tsukada, and H. Matsutani, "An on-device federated learning approach for cooperative model update between edge devices," *IEEE Access*, vol. 9, pp. 92986–92998, 2021.

[12] Y. Liu, N. Kumar, Z. Xiong, W. Y. B. Lim, J. Kang, and D. Niyato, "Communication-efficient federated learning for anomaly detection in industrial internet of things," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pp. 1–6, 2020.

[13] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, “Internet of things intrusion detection: Centralized, on-device, or federated learning?,” *IEEE Network*, vol. 34, no. 6, pp. 310–317, 2020.

[14] T. Zhang, C. He, T. Ma, L. Gao, M. Ma, and S. Avestimehr, “Federated learning for internet of things,” in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pp. 413–419, 2021.

[15] M. M. Rashid, S. U. Khan, F. Eusufzai, M. A. Redwan, S. R. Sabuj, and M. Elsharief, “A federated learning-based approach for improving intrusion detection in industrial internet of things networks,” *Network*, vol. 3, no. 1, pp. 158–179, 2023.