FL-IDS: Federated Learning-Based Intrusion Detection System Using Edge Devices for Transportation IoT

Group: **NEXUS**

Md. Siddiqur Rahman Tanveer (g202417180)
Asrafuzzaman Khan Nahin (g202318990)
Nafisa Tabassum (g202424600)

King Fahd University of Petroleum and Minerals
Department of Computer Engineering

COE 550: Internet of Things: Applications and Implementation
Term: **242**

FL-IDS: Federated Learning-Based Intrusion Detection System Using Edge Devices for Transportation IoT

Md. Siddiqur Rahman Tanveer (g202417180)
Asrafuzzaman Khan Nahin (g202318990)
Nafisa Tabassum (g202424600)
King Fahd University of Petroleum and Minerals
Computer Engineering Department
COE 550: IoT: Applications and Implementation – Term 242

Abstract— **This project introduces a Federated Learning-based Intrusion Detection System (FL-IDS) in IoT edge device environments, designed to improve the security of vehicular networks. The FL-IDS protects user data by training models locally on devices. Instead of sharing raw data, only model updates are sent to a central server, which combines them to create a stronger detection model. This approach helps maintain privacy. The system uses techniques like Logistic Regression (LR) and Convolutional Neural Networks (CNN) to detect and prevent attacks in transportation-related IoT networks. To test performance between these two algorithms, we have used one dataset: NSL-KDD. The model's performance was measured using accuracy and loss metrics. In this work, we have achieved higher accuracy in the PCC-CNN (78.50%) model than the Logistic Regression model (77.41%). In terms of loss, PCC-CNN gave a higher loss (1.087) than logistic regression (0.703).**

I. Introduction

The Internet of Things (IoT) is expanding at a rapid pace. IoT networks are currently using a large number of smart devices, including smartphones, smart automobiles, and smart apps. Transportation is one of the most significant and challenging applications of IoT, particularly in terms of Connected and Autonomous Vehicles (CAVs). The goals of these cars is to decrease accidents and make roadways safer. IoT devices related to transportation are mostly focused on self-driving cars. A Boston Consulting Group analysis projects that by 2025, the market for autonomous vehicles is expected to grow to $42 billion [1].

However, ensuring the safety and reliability of these vehicles is one of the most difficult tasks. Because there are many electronic components used, and internal and external communication systems make the system more complex. Therefore, it is crucial to create robust Intrusion Detection Systems (IDS) in order to identify and prevent various types of intrusions. In addition to security, CAV systems must be quick, smart, and capable of doing complex computations, which presents still another challenge. The ability of connected and autonomous vehicles (CAVs) to gather and analyze their environment by gathering data from sensors and transmitting it to other CAVs over wireless networks has generated a lot of studies [2].

Although effective, traditional centralized machine learning-based intrusion detection techniques usually require large-scale data aggregation at a central server, exposing private data and facing latency and bandwidth issues. On the other hand, Federated Learning (FL) enables the training of machine learning models without the need to share data with a central server. Rather, learning takes place directly on edge devices, such as sensors or cellphones. A central server receives only the model updates, not the real data like in figure 1. Because it retains sensitive information on the device and prevents it from

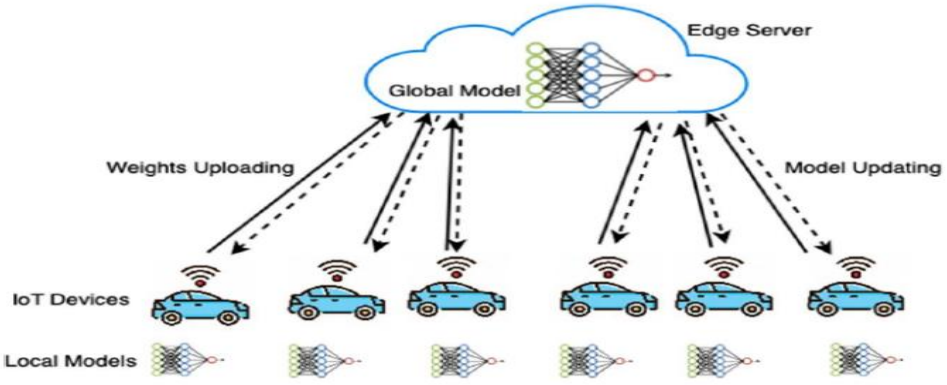

Fig. 1. Federated Learning Framework

being accessed, this technique is highly helpful when security and privacy are crucial.

FL has received significant interest in domains with distributed and privacy-sensitive data sources, such as IoT networks, healthcare, mobile systems, and, particularly, automotive networks [3]. In vehicular situations, making quick decisions based on sensor data is critical for ensuring safety and increasing transportation efficiency. Vehicles equipped with sensors like cameras, radars, Lidars, and GPS modules rely heavily on real-time data processing to help with navigation, collision avoidance, and advanced driver assistance systems (ADAS) [3].

Furthermore, two major issues need to be resolved in CAV systems: (1) the requirement for high bandwidth and low latency to transfer massive amounts of sensor data to centralized servers, and (2) the necessity of balancing local and global data to make the best decisions [4], [5]. While relying just on local data may limit a vehicle's capacity to make globally optimal decisions, transmitting massive datasets in real time can put excessive stress on vehicular networks and cause unacceptable delays. Therefore, a method that considers both local and global information while facilitating distributed learning is crucial.

In this work, we propose a lightweight Federated Learning-based Intrusion Detection System (FL-IDS) designed for transportation IoT networks. Unlike previous studies that relied on high-end embedded devices such as NVIDIA Jetson Xavier, our architecture exclusively utilizes Raspberry Pi 4 devices as both clients and the aggregator for the 1st testbed, and in another testbed, we have used our PC as an aggregator. The IDS models are trained and evaluated using the publicly available NSL-KDD dataset, a widely recognized benchmark for network intrusion detection.

II. Existing solution

In paper [6], they used federated learning so that different devices can use their local data to help train a model without compromising data privacy and security. Their FL framework is strong and independent of network type, which makes it simple to employ in various IoT systems and car networks. They compared the performance of intrusion detection between two popular datasets. Moreover, their system is flexible, as it can easily grow to support many users. They achieved almost 96.82% accuracy using the NSL-KDD dataset along with a loss of 0.127.

The paper also discussed several types of attacks that happened in the past. Based on the surveys [7]-[10], the most common attacks are DoS, DDoS, brute force, integrity, sniffing, Fuzzy, malware, and web. Many techniques have been identified to detect and overcome these security threats. These days, IDS has demonstrated encouraging outcomes in this regard [11]–[14]. Future research directions for developing and enhancing Intrusion Detection Systems (IDS) for Mobile Ad Hoc Networks (MANETs) and Vehicular Ad Hoc Networks (VANETs) were also highlighted by the review studies in [15] and [16]. Additionally, they emphasized the significance of protecting Internet of Things (IoT) device security.

III. Proposed Solution

According to our experimental findings, the suggested FL-IDS framework attains accuracy levels that are comparable to those in the paper. It confirms that implementing resource-efficient, privacy-preserving IDS solutions on low-power edge devices is feasible. The following is a summary of this paper's contributions:

1. We design a fully decentralized FL-IDS architecture using Raspberry Pi 4 devices as IoT clients.
2. We demonstrate that detection can be achieved even on constrained hardware, making the solution practical for widespread vehicular IoT deployments.
3. We have used PCC-CNN and the Logistic Regression (LR) algorithm separately to evaluate and compare performance.
4. We evaluate the system using the NSL-KDD dataset, showcasing the model's robustness and efficiency in a real-world vehicular security context.
5. We contribute to the growing body of research focused on lightweight, privacy-preserving, and scalable cybersecurity solutions for transportation IoT infrastructures.

The main disadvantage of our project is that we couldn’t achieve higher accuracy compared to the existing paper. Moreover, our loss is higher than the previously proposed model. The reason behind this will be discussed further in the results analysis section.

IV. Design Methodology

Discuss your design and justify your design choices. Discuss system architecture, network architecture, computing architecture, and protocols/technologies of choice.

A. Network Architecture

Figure 2 illustrates our suggested FL-based IDS system architecture. [6] is the source of this inspiration. The FL environment was implemented using the Flower framework, and anomalies were detected using the CNN-IDS approach.

Flower [17] is a tool that helps to build and run full Federated Learning (FL) models. It gives different ways to combine results (aggregation algorithms) and lots of options to customize for real-world devices. Flower works well both in

simulations and when used on actual devices [17]. The Flower framework has two main parts: the global model and the local model. Each client (or device) trains its local model and then shares its model updates to improve the global model. The global model is in charge of picking which clients to use, combining their updates, setting configurations, and checking how well the model is doing. It can do all this in a distributed or centralized way using something called strategy abstraction. The FedAvg algorithm is used for this purpose.

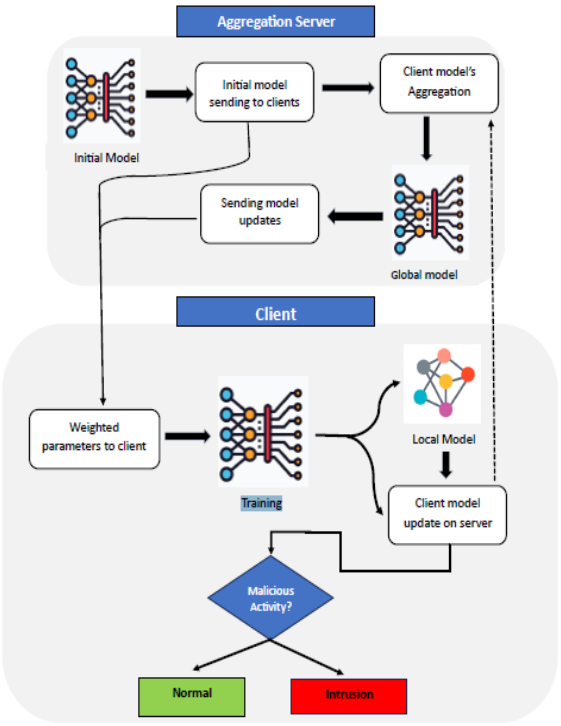

Fig. 2. Proposed FL-IDS Framework

B. System Architecture

To design the testbed, we used a Python file for the server model, separate Python files for each client model, and another Python file for data preprocessing. Based on the specific setup, the testbed ran on a powerful PC as the server and Raspberry Pi 4 devices as the clients and server.

TABLE 1. SPECIFICATIONS OF TESTBED ENVIRONMENT

Feature	Raspberry Pi 4 B
SoC	Quad-core ARM Cortex-A72 64-bit
RAM	8 GB
Storage	MicroSD 32 GB
OS	Debian Bookworm 64 bit

Testbed: Initially, we set two clients for testing as in Fig.3. This setup consisted of one Raspberry Pi 4 as the server and two clients as two Raspberry Pi 4. For decentralized machine learning, we employed PCC-CNN and logistic regression (LR) on NSL-KDD datasets. Two algorithms were run independently by each client to analyze the performance.

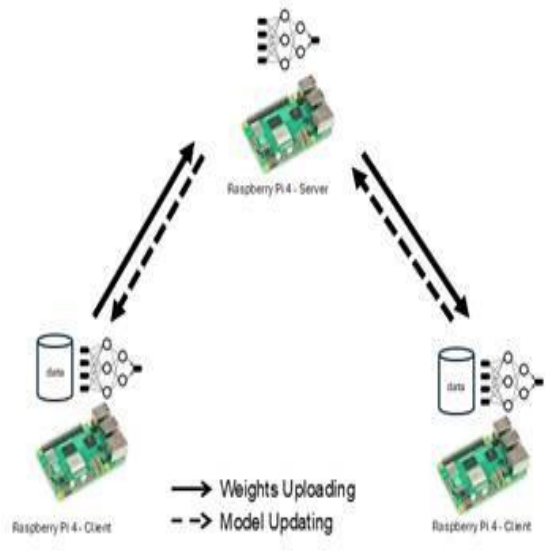

Fig. 3. FL testbed with two clients

Datasets: The NSL-KDD dataset [18], a popular standard for network traffic analysis, was used here for evaluation. It was collected by the University of New Brunswick (UNB) and is considered more realistic because the data samples were not generated with the same probability, avoiding repetition.
The main focus of the dataset is on DDoS (Distributed Denial of Service) attacks. It comes with two CSV files: KDD_Train and KDD_Test, which are used for training and testing machine learning models. The NSL-KDD dataset includes five types of traffic classes:

1. DoS (Denial of Service): These attacks try to flood a server or network with too many requests, making it crash or become unresponsive.
2. Probe: These are scanning attacks where the attacker tries to gather information about the network to find weak points for future attacks.
3. R2L (Remote to Local): In this type, the attacker tries to gain access to a system from a remote location by sending malicious packets.
4. U2R (User to Root): The attacker starts as a regular user and tries to gain root or admin privileges by exploiting system vulnerabilities.
5. Normal: This represents regular, non-malicious network traffic.

The attack instances of NSL-KDD are shown in Table 2.

TABLE 2. NSL-KDD ATTACK SAMPLES

Attack Category	Attack Instances	
	NSL_Train	NSL_Test
Normal	67343	9711
DOS	45927	7460
Probe	11656	2885
R2L	995	2421
U2R	52	67

C. Computation Architecture

In this work, we have investigated 2 types of computation performed through logistic-regression and PCC-CNN-based IDS.

Logistic-Regression based IDS: The model is a simple logistic regression classifier implemented using TensorFlow's Keras API. It takes input data with shape (features, 1), which is first flattened into a 1D vector using a Flatten layer. It is appropriate for binary classification and is followed by a single Dense output layer with two units and a SoftMax activation function. The model does not include any hidden layers, reflecting the structure of traditional logistic regression. It is compiled using the Adam optimizer with a low learning rate (0.0001), categorical cross-entropy loss, and accuracy as the performance metric.

PCC-CNN based IDS: This model is a 1D Convolutional Neural Network (CNN) designed for classification tasks involving sequential or structured data, such as network traffic in the NSL-KDD dataset. It begins with two stacked 1D convolutional layers, each with 64 filters and a kernel size of 10, both using the ReLU activation function and L2 regularization to prevent overfitting. These layers learn spatial patterns across the input features. A MaxPooling1D layer follows to downsample the feature maps, reducing dimensionality and computation. The output is then flattened into a vector, allowing transition into a fully connected layer with 128 neurons and ReLU activation, again regularized with L2. To further prevent overfitting, a Dropout layer (30%) is included, which randomly disables some neurons during training. The class probabilities are then produced by a Dense layer with two units and SoftMax activation, which is appropriate for binary classification with one-hot encoded labels. With categorical cross-entropy as the loss function, accuracy as the evaluation metric, and a small learning rate (0.0001) for stable training, the model is assembled using the Adam optimizer.

D. Technologies and Protocols

TABLE 3. TECHNOLOGY/PROTOCOLS USED IN EACH LAYER

Layer	**Technology / Protocol**	**Purpose**
Networking	Wi-Fi (802.11)	Connectivity
Communication	HTTP	Client-server communication
Transport Layer	TCP/IP	Reliable data transmission
Application Layer	REST	Client-aggregator data exchange
ML Framework	TensorFlow	Model definition, training, inference
Federated Strategy	FedAvg	Model update aggregation
Web Server	Flask (Python)	Hosting REST endpoints

Table 3 shows all the protocols and technologies used in our project for each layer.

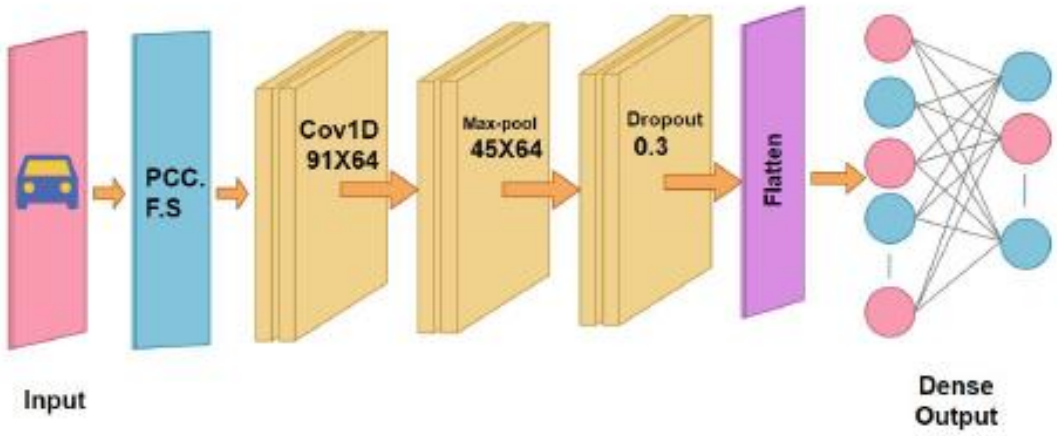

Fig. 4. FL testbed with two clients

V. PERFORMANCE EVALUATION

In the federated learning approach, the performance is evaluated through loss and accuracy metrics. These two metrics have been calculated in both local clients and the global aggregator.

A. *Performance Metrics*

In the federated learning approach, the performance is evaluated through loss and accuracy metrics. These two metrics have been calculated in both local clients and the global aggregator.

Loss: The objective of training a model in machine learning is to reduce the prediction error. To accomplish this, models employ a loss function, which determines the degree to which the model's forecasts and actual values agree. The model performs better if the loss function values get smaller. Because cross-entropy is a good way to measure a classification model's performance, it is frequently used for classification tasks. The commonly used loss function is Categorical Cross-Entropy Loss, as mathematically represented below:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \cdot \log(\hat{y}_{ij}) \quad (1)$$

where,
N: Number of samples in the dataset (local or global)
C: Number of classes
y_{ij}: Ground truth (1 if sample i belongs to class j, else 0)
$\hat{y}_{ij}$: Predicted probability that sample i belongs to class j.

In federated learning, we have used local loss computed on the client's local data and the global loss computed on a centralized test dataset or averaged from local losses (weighted by dataset size):

$$L_{global} = \frac{1}{\sum_{k=1}^{k} n_k}\sum_{k=1}^{k} n_k . L_k \quad (2)$$

where,
K: Number of clients
n_k: Number of samples on client k
L_k: Loss computed by client k

Accuracy: Accuracy measures how often the predicted label matches the true label. Mathematically

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

$$= \frac{1}{N}\sum_{i=1}^{N} 1.(\hat{y}_i = y_i) \quad (3)$$

$\hat{y}_i$: Predicted class label for sample i
y_i: True class label for sample i
$1(\cdot)$: Indicator function that returns 1 if the condition is true, 0 otherwise
N: Number of samples

In Federated Learning, Local accuracy is computed on local client data. Global accuracy computed on a centralized test set or weighted sum from clients:

$$Accuracy_{global} = \frac{1}{\sum_{k=1}^{k} n_k}\sum_{k=1}^{k} Correct_k \quad (4)$$

$Correct_k$: Number of correct predictions by client k

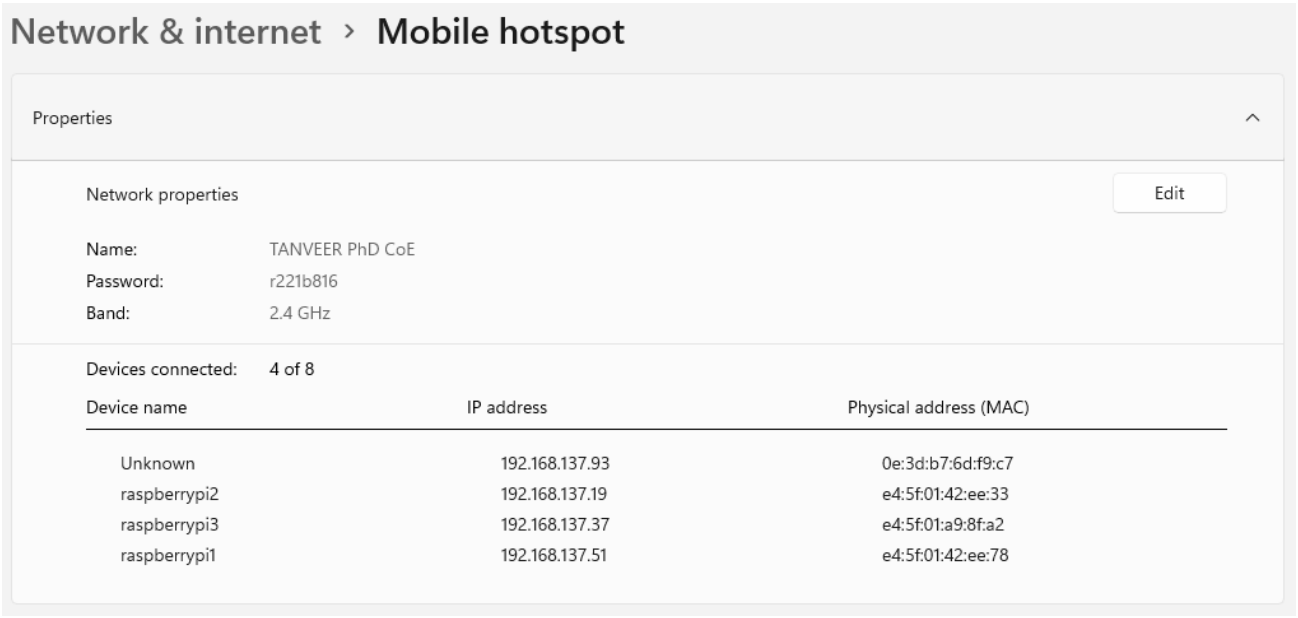

Fig. 5. Wi-Fi Access point

B. *Real-time Implementation*

In this work, we have implemented federated learning on a Raspberry Pi 4 B. We have used 1 Raspberry Pi as an aggregator and 2 other Raspberry Pis as local clients. For wireless communication among the IoT devices, we have used the Wi-Fi mobile hotspot in our laptop, as it worked as an access point. The laptop is also sharing the internet, its own Internet, alongside the hotspot to the Raspberry Pi devices. As a result, we update the Raspberry Pi devices to the latest packages up to date. The following description shows the step-by-step implementation on real devices.

Wi-Fi Hotspot creation: In our laptop, we have shared our internet with the Raspberry Pi devices through the mobile hotspot option provided by Windows 11 OS installed in the laptop. We have created a Wi-Fi access point "TANVEER PhD CoE" and the Raspberry Pi devices got connected to this access point, and the access point provided the IP addresses to the devices through DHCP shown in Fig.5.

IP address allocation to Raspberry Pi devices: As shown in Fig.4, we can see that our Raspberry Pi 1 (e4:5f:01:42:ee:78), which we considered as our aggregator to our system, got IP address 192.168.137.51. And the other client 1 that is raspberrypi2 (e4:5f:01:42:ee:33) obtained 192.168.137.19 and the client 2 (e4:5f:01:a9:8f:a2) got 192.168.137.37.

Raspberry Pi remote desktop connection: We installed the Debian Bookworm Linux kernel-based operating system to Raspberry Pi devices, which included the remote desktop features so that we can access the Raspberry Pi desktop remotely instead of using the SSH command application like PuTTY. That's why we were able to transfer the necessary files to the Raspberry Pi devices. That's why we used the RealVNC Viewer remote desktop viewer to interact with the Raspberry Pi devices represented in Fig. 6.

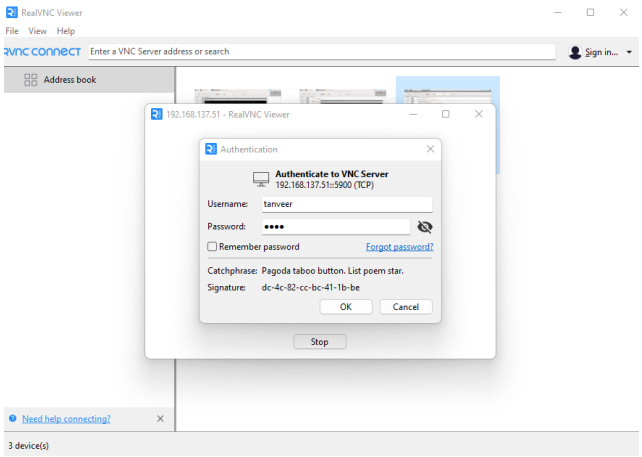

Fig. 6. Raspberry pi remote desktop access

File transfer to raspberry from PC: We used the windows command terminal to transfer the files such as (python scripts and dataset csv files) to the aggregator and clients shown in Fig.7

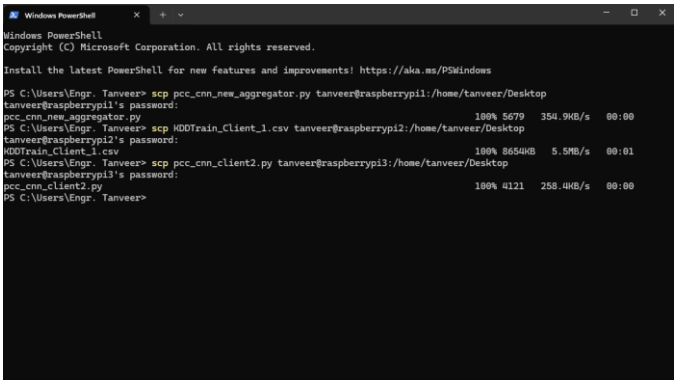

Fig. 7. File transfer to Raspberry Pi devices.

Initiating the aggregator server: Since our aggregator raspberry pi device is now ready then we initiated an HTTP server on the raspberry pi aggregator.

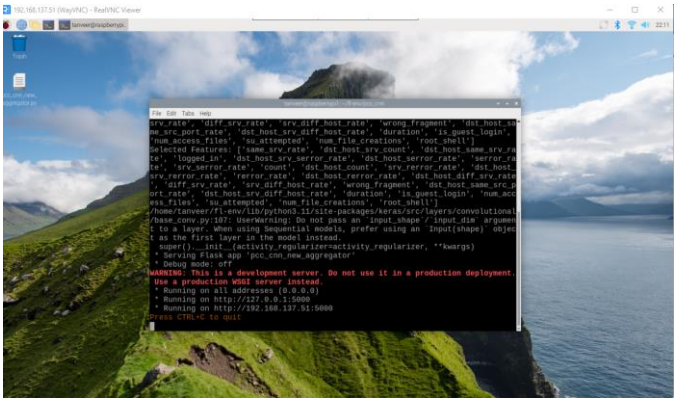

Fig. 8. Aggregator server started at IP 192.168.137.51.

In Fig. 8, we can see that the server is now waiting for the clients to connect.

Connect the clients to the aggregator: We have updated the code in the client scripts to the aggregator URL 192.168.137.51 so that when we run the Python scripts, it will get connected to the server and through the client–server communication model, they can communicate and exchange data between them.

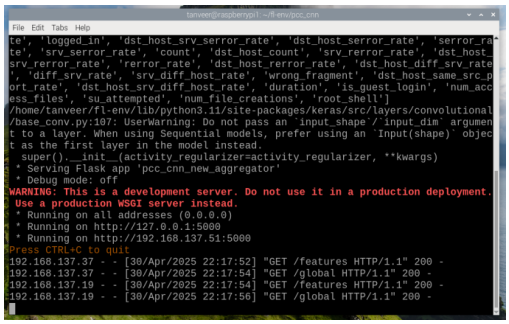

Fig. 9. Raspberry Pi clients connected to the aggregator

In Fig. 9 we can see that our local clients got connected to the global aggregator

Sending initial global model to clients: In Fig. 10, at first, the global initial model weights were sent to the local clients.

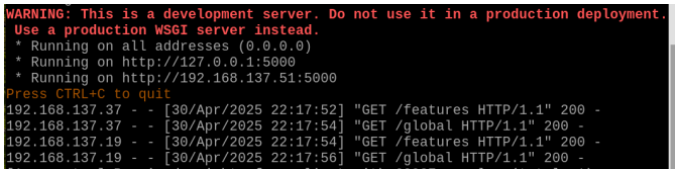

Fig. 10. Global weight initially sent to connected clients

Local training on clients: As soon as the clients get the features and initial global weights, they start training their dataset.

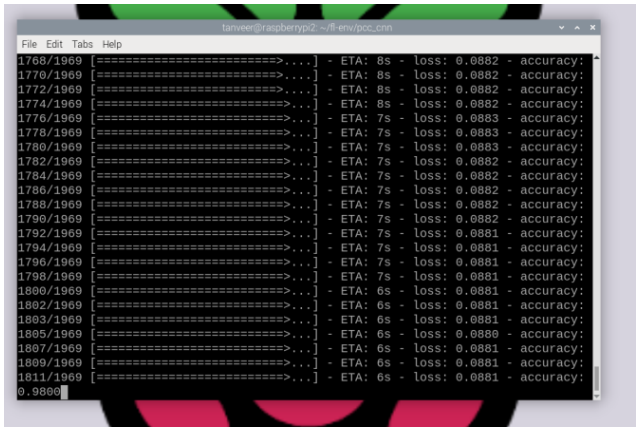

Fig. 11. Local training on client 1

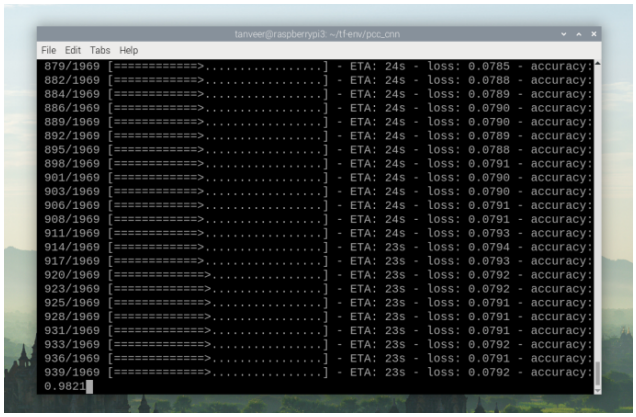

Fig. 12. Local training on client 2

In Fig. 11-12, we can see that the clients are training their dataset and showing the corresponding local loss and accuracy.

Local weights aggregation on server: As soon as the client completes its training, it instantly sends the local weights to the aggregator through the REST API. The aggregator is always listening to the clients, and when it receives the local weights from the clients, it aggregates and sends back the global weight to the client shown in Fig.13.

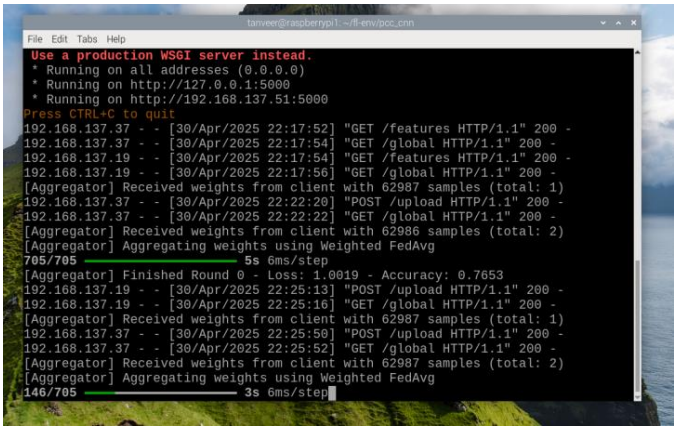

Fig. 13. Aggregation on the aggregator and weights fed back to clients.

Saving the logs in global aggregator: We saved the global weights in each round in a.pkl file, shown in Fig. 14

Fig. 14. Global weights saved in each round.

Also, we stored the training history in a training_history.csv file so that we can understand in which round there caused any error while aggregation, like in Fig. 15

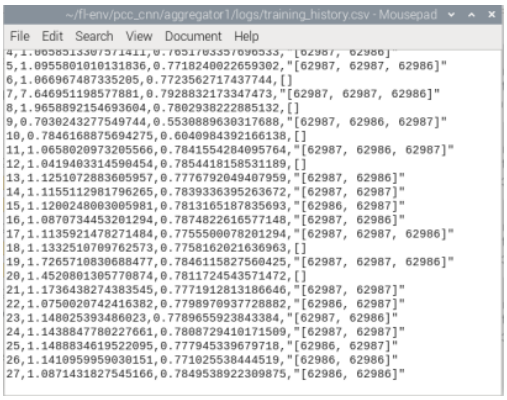

Fig. 15. Training history of each round

Fig. 16. Actual implementation of our model

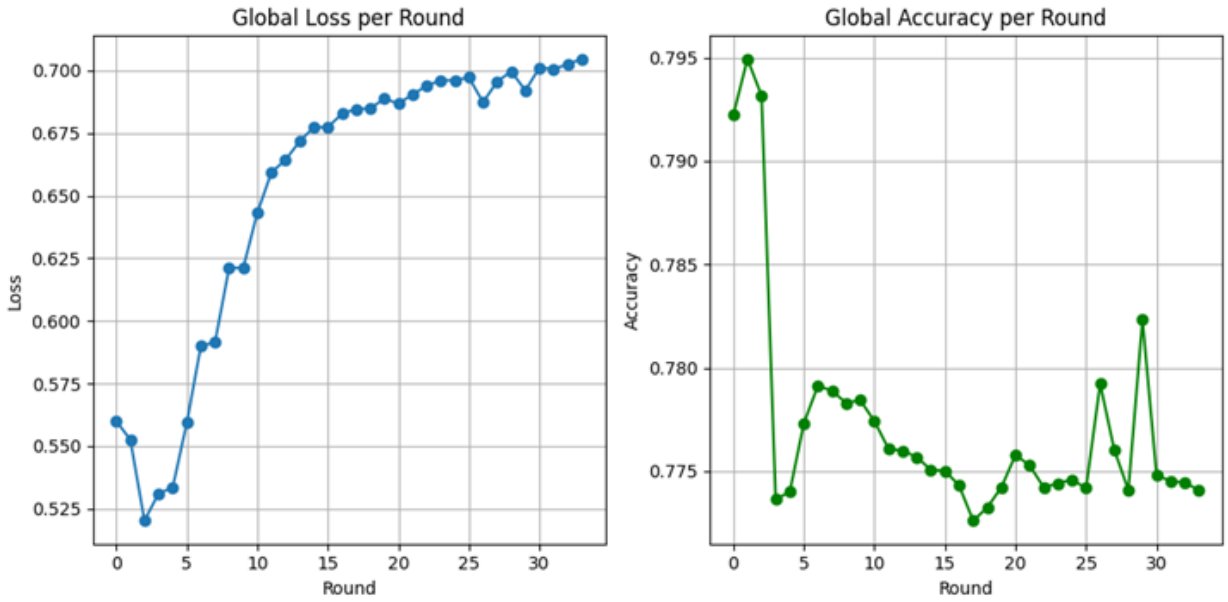

Fig. 17. Performance per round with Logistic Regression

Finally, the visual implementation is given in fig. 16

C. Experimental Results and Analysis

Fig-17 and 18 show how our system is responding to detect anomalies. As we can see, with each round, the loss is increasing, which is making the accuracy decrease. For logistic regression, we got almost 77.41% accuracy with 0.703 loss. On the other hand, for PCC-CNN, we got 78.5% accuracy with 1.087 loss. For both algorithms, we achieved higher loss and lower accuracy compared to the existing model.

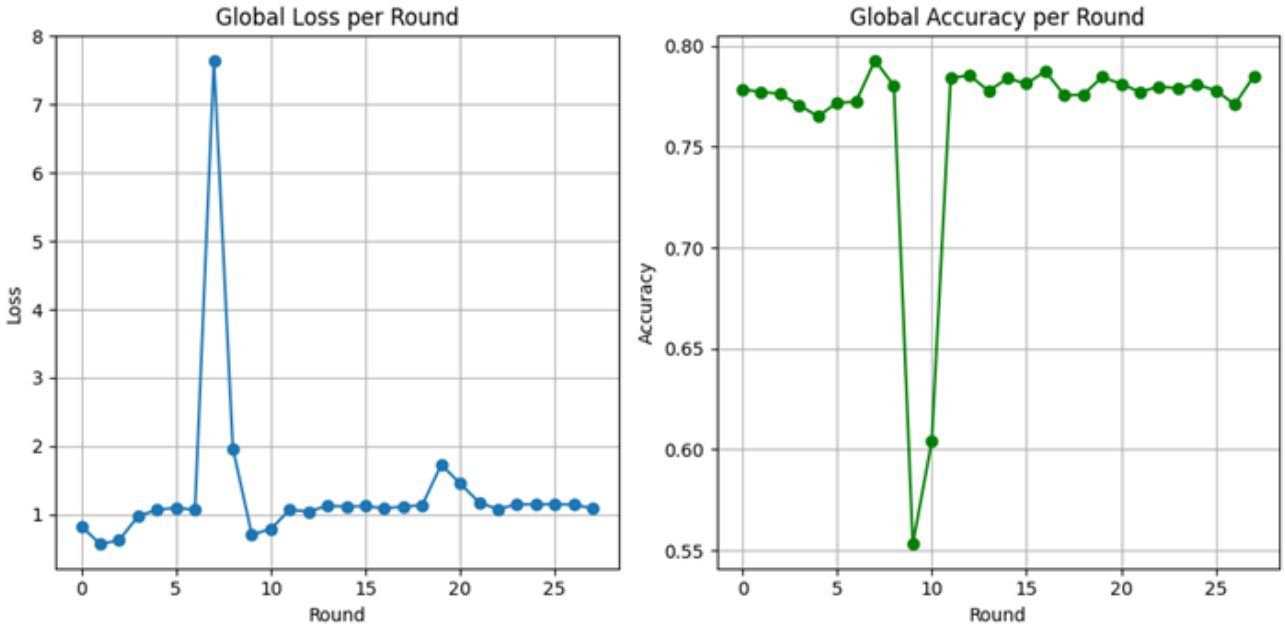

Fig. 18. Performance per round with PCC-CNN

In tables 4 and 5, we are presenting how much loss and accuracy we are getting in each round for sample counts in the Logistic regression model and PCC-CNN model, respectively. As you can see, for Logistic regression, there were 33 rounds, whereas for PCC-CNN, there were 27 rounds. And, Table 6 shows the summary of our results with two models.

TABLE 4. LOSS AND ACCURACY PER ROUND IN LOGISTIC REGRESSION

Round	Loss	Accuracy	SampleCounts
0	0.56003	0.792229	[62986, 62987, 62986]
1	0.552547	0.794934	[]
2	0.520205	0.79316	[62987, 62986]
3	0.531136	0.773643	[62987, 62986, 62987]
4	0.533433	0.773998	[]
5	0.559261	0.777324	[62986, 62987]
6	0.590074	0.779143	[62986, 62987, 62986]
7	0.59144	0.778877	[]
8	0.621228	0.7783	[62987, 62986, 62987]
9	0.621145	0.778433	[]
10	0.643215	0.777413	[62986, 62987]
11	0.659372	0.776082	[62986, 62987]
12	0.664337	0.775994	[62986, 62987]
13	0.671743	0.775683	[62986, 62987]
14	0.67735	0.775062	[62986, 62987, 62986]
15	0.677424	0.775018	[]
16	0.682747	0.774308	[62987, 62986]
17	0.68466	0.772605	[62987, 62986, 62987]
18	0.684861	0.773246	[]
19	0.688745	0.774219	[62987, 62986]
20	0.686875	0.775785	[62987, 62986, 62987]
21	0.690389	0.775279	[]
22	0.693972	0.774219	[62987, 62986]
23	0.696119	0.774412	[62987, 62986, 62987]
24	0.696142	0.774594	[]
25	0.69725	0.774175	[62987, 62986]
26	0.687342	0.779229	[62987, 62987, 62986]
27	0.695638	0.776036	[]
28	0.699671	0.774086	[62987, 62986]
29	0.692042	0.782328	[62987, 62987, 62986]
30	0.701218	0.774792	[]
31	0.700608	0.77453	[62987, 62986]

Round	Loss	Accuracy	SampleCounts
32	0.702588	0.774441	[62986, 62986]
33	0.704651	0.774086	[62986, 62986]

TABLE 5. LOSS AND ACCURACY PER ROUND IN PCC-CNN

Round	Loss	Accuracy	Sample Counts
0	0.824725	0.778478	[62987, 62986]
1	0.564123	0.777243	[62987, 62986, 62987]
2	0.622278	0.776264	[]
3	0.976257	0.770538	[62987, 62986]
4	1.065851	0.76517	[62987, 62986]
5	1.09558	0.771824	[62987, 62987, 62986]
6	1.066967	0.772356	[]
7	7.646951	0.792883	[62987, 62987, 62986]
8	1.965889	0.780294	[]
9	0.703024	0.553089	[62987, 62986, 62987]
10	0.784617	0.604098	[]
11	1.065802	0.784155	[62987, 62986, 62987]
12	1.04194	0.785442	[]
13	1.125107	0.777679	[62987, 62986]
14	1.115511	0.783934	[62987, 62987]
15	1.120025	0.781317	[62986, 62987]
16	1.087073	0.787482	[62987, 62986]
17	1.113592	0.77555	[62987, 62987, 62986]
18	1.133251	0.775816	[]
19	1.726571	0.784612	[62987, 62987, 62986]
20	1.45208	0.781172	[]
21	1.173644	0.777191	[62987, 62987]
22	1.075002	0.779897	[62986, 62987]
23	1.148025	0.778966	[62987, 62986]
24	1.143885	0.780873	[62987, 62987]
25	1.148883	0.777945	[62986, 62987]
26	1.141096	0.771026	[62986, 62986]
27	1.087143	0.784954	[62986, 62986]

TABLE 6. PERFORMANCE ANALYSIS FOR IDS SYSTEM

Algorithm	Round	Epoch	Accuracy	Loss
PCC-CNN	27	5	78.50	1.087
Logistic Regression	33	1	77.41	0.703

D. *Performance Comparison*

If we compare our results with existing paper shown in Table 7, we can observe that we have gained less accuracy and more loss than them. By using the PCC-CNN model and two clients, they have got 96.82%, whereas we got only 78.5% with a high loss factor.

TABLE 7. PERFORMANCE COMPARISON BETWEEN OUR DESIGNED MODEL AND THE PROPOSED MODEL

Algorithm	Proposed Model		Designed Model	
	Accuracy	Loss	Accuracy	Loss
Logistic regression	96.82	0.127	77.41	0.703
PCC-CNN	96.82	0.127	78.5	1.087

Again, while using logistic regression, we got 77.41% accuracy, which is again much lower than their accuracy (96.82%). The reasons behind this are as follows:

- Dataset sample distribution
- Feature selection criteria were not mentioned in the paper
- Not using the Jetson Xavier server
- Imbalance between clients: One client finished first, while the other client was still training. That's why global weights get very imbalanced. And for the same reason, weights between two clients were
- A low number of clients and a dataset split required a better sample distribution

VI. CONCLUSIONS

We have implemented an IDS system using PCC-CNN and logistic regression. The paper that we followed to reproduce [6] showed much higher accuracy than our designed model. Initially, we used fewer IoT clients and a more constrained server than them. We tried to get better accuracy, but we failed to do so because of some limitations and drawbacks of our design. In the paper [6], they used Jetson Xavier as a server, which is more powerful than Raspberry Pi when it comes to working as a server. But, we took a challenge to use a more constrained device such as Raspberry Pi 4 B, which has a significant impact on our performance. Additionally, the way they have split the datasets is not clearly mentioned in the paper. For that reason, we don't know which features they selected using the Pearson Correlation Coefficient (PCC) from the datasets. Moreover, the threshold is not mentioned; that's why we couldn't cross-validate our performance with them. In the future, we can include more IoT clients to get better accuracy, as well as we can upgrade our models further for better analysis of datasets.

REFERENCES

[1] BCG: Autonomous Car Market to Hit 42 Billion by 2025. Accessed: May 5, 2023. [Online]. Available: https://www.consultancy.uk/news/2065/bcg-autonomous-car-market-to-hit-42-billion-by-2025

[2] M. R. Jabbarpour, A. Nabaei, and H. Zarrabi, ‘‘Intelligent guardrails: an iot application for vehicle traffic congestion reduction in smart city,’’ in *2016 IEEE international conference on Internet of Things (Ithings) andIEEE Green computing and communications (Greencom) and IEEE cyber,physical and social computing (cpscom) and IEEE smart data (smartdata)*.IEEE, 2016, pp. 7–13.

[3] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: An intrusion detection case study," Appl. Sci., vol. 8, no. 12, p. 2663, Dec. 2018. [Online]. Available: https://www.mdpi.com/2076-3417/8/12/2663

[4] S. Savazzi, M. Nicoli, M. Bennis, S. Kianoush, and L. Barbieri, "Opportunities of federated learning in connected, cooperative, and automated industrial systems," IEEE Commun. Mag., vol. 59, no. 2, pp. 16–21, Feb. 2021.

[5] Z. Du, C. Wu, T. Yoshinaga, K. A. Yau, Y. Ji, and J. Li, "Federated learning for vehicular Internet of Things: Recent advances and open issues," IEEE Open J. Comput. Soc., vol. 1, pp. 45–61, 2020.

[6] M. H. Bhavsar, Y. B. Bekele, K. Roy, J. C. Kelly and D. Limbrick, "FL-IDS: Federated Learning-Based Intrusion Detection System Using Edge Devices for Transportation IoT," in IEEE Access, vol. 12, pp. 52215-52226, 2024

[7] K. Kim, J. S. Kim, S. Jeong, J.-H. Park, and H. K. Kim, ‘‘Cybersecurity for autonomous vehicles: Review of attacks and defense,’’ Computers & Security, vol. 103, p. 102150, 2021.

[8] J. Cui, L. S. Liew, G. Sabaliauskaite, and F. Zhou, ‘‘A review on safety failures, security attacks, and available countermeasures for autonomous vehicles,’’ Ad Hoc Networks, vol. 90, p. 101823, 2019

[9] X. Sun, F. R. Yu, and P. Zhang, ‘‘A survey on cyber-security of connected and autonomous vehicles (cavs),’’ IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 7, pp. 6240–6259, 2021.

[10] D. Man, F. Zeng, J. Lv, S. Xuan, W. Yang, and M. Guizani, ‘‘Ai-based intrusion detection for intelligence internet of vehicles,’’ IEEE Consumer Electronics Magazine, vol. 12, no. 1, pp. 109–116, 2023.

[11] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D. Gan, ‘‘Cloud-based cyber-physical intrusion detection for vehicles using deep learning,’’ Ieee Access, vol. 6, pp. 3491–3508, 2017.

[12] A. Tomlinson, J. Bryans, and S. A. Shaikh, ‘‘Towards viable intrusion detection methods for the automotive controller area network,’’ in 2nd ACM Computer Science in Cars Symposium, 2018, pp. 1–9.

[13] V. Hajisalem and S. Babaie, ‘‘A hybrid intrusion detection system based on abc-afs algorithm for misuse and anomaly detection,’’ Computer Networks, vol. 136, pp. 37–50, 2018.

[14] I. Berger, R. Rieke, M. Kolomeets, A. Chechulin, and I. Kotenko, ‘‘Comparative study of machine learning methods for in-vehicle intrusion detection,’’ Springer International Publishing, pp. In InternationalWorkshop on Security and Privacy Requirements Engineering (pp. 85–101), 01 2019.

[15] S. Pamarthi and R. Narmadha, ‘‘Literature review on network security in wireless mobile ad-hoc network for iot applications: Network attacks and detection mechanisms,’’ International Journal of Intelligent Unmanned Systems, vol. 10, no. 4, pp. 482–506, 2022.

[16] M. Chowdhury, M. Islam, and Z. Khan, ‘‘Security of connected and automated vehicles,’’ arXiv preprint arXiv:2012.13464, 2020.

[17] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão et al., ‘‘Flower:Afriendly federated learning research framework,’’ arXiv preprint arXiv:2007.14390, 2020.

[18] M. Tavallaee, E. Bagheri,W. Lu, and A. A. Ghorbani, ‘‘A detailed analysis of the kdd cup 99 data set,’’ in *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 2009, pp. 1–6.