future internet

MDPI

Article

Hierarchical Federated Learning-Based Intrusion Detection for In-Vehicle Networks

Muzun Althunayyan [1,2,*], Amir Javed [1,*], Omer Rana [1] and Theodoros Spyridopoulos [1]

[1] School of Computer Science and Informatics, Cardiff University, Cardiff CF10 3AT, UK; ranaof@cardiff.ac.uk (O.R.); spyridopoulost@cardiff.ac.uk (T.S.)
[2] Computer Sciences and Information Technology College, Majmaah University, Al Majma'ah 11952, Saudi Arabia
* Correspondence: althunayyanms@cardiff.ac.uk (M.A.); javeda7@cardiff.ac.uk (A.J.)

Abstract: Intrusion detection systems (IDSs) are crucial for identifying cyberattacks on in-vehicle networks. To enhance IDS robustness and preserve user data privacy, researchers are increasingly adopting federated learning (FL). However, traditional FL-based IDSs depend on a single central aggregator, creating performance bottlenecks and introducing a single point of failure, thereby compromising robustness and scalability. To address these limitations, this paper proposes a Hierarchical Federated Learning (H-FL) framework to deploy and evaluate the performance of the IDS. The H-FL framework incorporates multiple edge aggregators alongside the central aggregator, mitigating single-point failure risks, improving scalability, and efficiently distributing computational load. We evaluate the proposed IDS using the H-FL framework on two car hacking datasets under realistic non-independent and identically distributed (non-IID) data scenarios. Experimental results demonstrate that deploying the IDS within an H-FL framework can enhance the F1-score by up to 10.63%, addressing the limitations of edge-FL in dataset diversity and attack coverage. Notably, H-FL improved the F1-score in 16 out of 24 evaluated scenarios. By enabling the IDS to learn from diverse data, driving conditions, and evolving threats, this approach substantially strengthens cybersecurity in modern vehicular systems.

Keywords: CAN bus; cyberattack; IDS; federated learning; in-vehicle network

Citation: Althunayyan, M.; Javed, A.; Rana, O.; Spyridopoulos, T. Hierarchical Federated Learning-Based Intrusion Detection for In-Vehicle Networks. *Future Internet* **2024**, *16*, 451. https://doi.org/10.3390/fi16120451

Academic Editors: Olivier Markowitch and Jean-Michel Dricot

Received: 10 October 2024
Revised: 20 November 2024
Accepted: 27 November 2024
Published: 3 December 2024

1. Introduction

The number of connected and autonomous vehicles (CAVs) is experiencing rapid growth globally. This surge in reliance on electronics and software within modern automotive systems introduces cybersecurity risks that were not previously considered in their design and engineering [1]. A recent report estimates that the global market for connected vehicles will grow to approximately USD 244 billion by 2030 [2]. The controller area network (CAN) bus has emerged the primary protocol for in-vehicle communication because of its simplicity and efficiency. However, it lacks essential security features such as authentication and encryption [3], making it vulnerable to cyberattacks that can have severe consequences, including loss of life [4,5]. Intrusion detection systems (IDSs) are an effective method for detecting these threats within in-vehicle networks. Despite significant advancements in machine learning and deep learning for IDSs, challenges related to data privacy and communication efficiency remain. Federated learning (FL) offers a promising solution by enabling local model training while preserving the privacy of raw data [6]. FL addresses key limitations of centralized IDS approaches and is particularly well suited for in-vehicle networks for several compelling reasons:

- Privacy Preservation:Regulations such as the general data protection regulation (GDPR), California consumer privacy act (CCPA), personal information protection and electronic documents act (PIPEDA), and Brazilian general data protection law

(LGPD) safeguard sensitive data from unauthorized movement. FL enhances privacy by transmitting only learned parameters to the cloud server instead of raw data, thereby improving data protection [7].

- Reduced latency: By processing data locally and only sending model updates, FL reduces latency compared to traditional centralized approaches that require sending raw data to a central server [7].
- Compliance with guidelines: According to the 2020 guidelines of the International Telecommunication Union, an in-vehicle IDS must have the ability to regularly update its set of rules [8]. FL facilitates these updates by allowing continuous learning and model refinement.
- Adaptability to new attacks: FL enhances IDS adaptability to new, unseen attacks by updating local models with those trained on newly identified threats. These continuous updates ensure effective responses to evolving threats.
- Diverse driving scenarios: FL facilitates the training of a universal model that encompasses various driving scenarios, vehicle states, and driving behaviors.

Numerous studies have developed FL-based IDSs for in-vehicle networks [9–14]. However, they all utilize the standard FL architecture. To the best of our knowledge, this is the first study to introduce the H-FL framework for an in-vehicle IDS and to deploy and evaluate IDS performance within this architecture. The aim of this paper is to introduce and deploy our proposed IDS in [15] in the H-FL environment, evaluating its performance. The main contributions are summarized as follows:

- Deployment and evaluation of our proposed multistage IDS in an H-FL environment.
- Distribution of data across clients with three levels of non-IID data distributions, making the scenario more realistic.
- Performance evaluation of the IDS in the H-FL environment for both seen and unseen attack detection.
- Assessing IDS performance under different levels of non-IID data distribution.

The remainder of this paper is organized as follows. Section 2 provides context and background on FL, followed by a discussion of related work. Section 3 describes our overall methodology, followed by experimental results and evaluation in Section 4. Finally, Section 5 concludes the paper.

2. Background and Related Work

This section begins by providing background information on cloud-based FL and HFL, followed by a review of related works on in-vehicle IDSs deployed using an FL approach. It then examines non-IID data distributions, how these works allocated data across clients, and the limitations of existing studies.

2.1. Cloud-Based FL and Hierarchical FL

The standard cloud-based FL architecture comprises a cloud server and numerous clients, as shown in Figure 1. Clients (vehicles) download a global model from the cloud server, undergo multiple rounds of local training, and then send the model weights back to the server for aggregation. The iteration continues until the model achieves the desired level of accuracy. In cloud-based FL, the number of participating clients can reach into the millions [16], which can cause communication with the cloud server to become slow and unpredictable due to factors like network congestion, ultimately leading to inefficiencies in the training process [16,17]. In contrast, H-FL architecture consists of a cloud server, multiple edge servers, and numerous clients. In H-FL, clients update their local parameters and send them to the nearest edge server for edge aggregations, following a similar approach as in cloud-based FL. The difference is that after several rounds of edge aggregations, multiple edge servers send their parameters to a cloud server for cloud aggregation. This design allows more clients to participate in the framework. Moreover, H-FL significantly reduces the costly communication with the cloud by leveraging efficient

client–edge interactions, leading to considerable decreases in both runtime and the number of necessary local iterations [18].

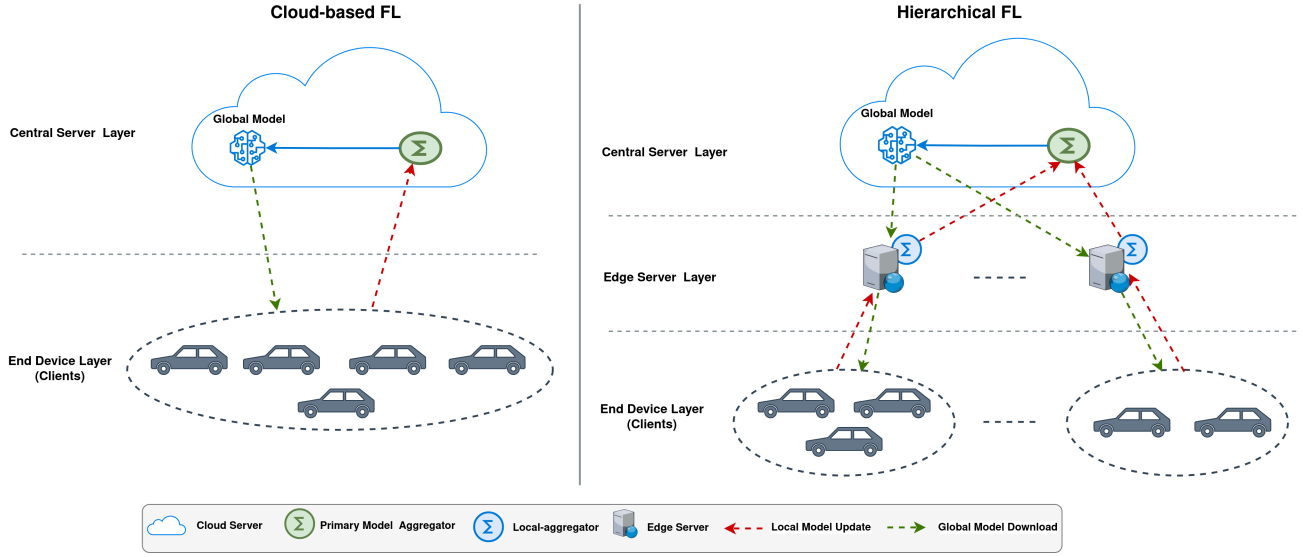


Figure 1. Cloud-based FL and H-FL.

2.2. Federated Learning-Based Intrusion Detection Systems

Driss et al. [9] introduced an FL-based framework to identify attacks in vehicular sensor networks. Given the resource constraints of smart sensing devices in vehicular networks, the authors emphasized the significance of adopting lightweight security approaches. To address this issue, they utilized a set of Gated Recurrent Units with an ensemble method based on Random Forest to aggregate the global machine learning models. They distributed the dataset equally among clients.

Shibly et al. [10] presented an FL-based IDS for enhancing security in autonomous systems. The authors personalized the model for each client participating in the FL process. The study integrates both supervised and unsupervised FL-based IDS to analyze patterns in CAN buses. Moreover, to create a more generalized model, local clients have the option to refine the global model using their local data and deploy it during inference.

Yu et al. [11] introduced an FL-based IDS based on Long Short-Term Memory (LSTM) for in-vehicle networks. They utilized the periodicity of CAN communications to predict the arbitration IDs of new messages. The 11-bit arbitration ID is transformed into vectors using one-hot encoding, and these vectors are utilized by the LSTM to predict the subsequent arbitration ID. Data are evenly distributed among clients, with each client holding 1000 instances for training and 200 instances for testing. A comparison between the FL-based and a centralized IDS revealed a 0.071 accuracy loss for the FL-based IDS. However, the authors suggested that this loss could be mitigated through a cumulative error scheme.

Zhang et al. [12] developed an anomaly detection system utilizing a graph neural network, capable of detecting CAN bus intrusions within a minimal 3-millisecond timeframe. They constructed a two-stage classifier cascade consisting of a classifier designed for anomaly detection in one class, and another classifier for categorizing attacks into multiple classes. To address novel anomalies from unseen classes, an openmax layer is integrated into the multi-class classifier.

Yang et al. [13] introduced an IDS for in-vehicle networks utilizing federated deep learning. The proposed approach leverages the periodicity of network messages, employs the ConvLSTM model, and trains the model through federated deep learning. To simulate a non-IID environment, clients were assigned varying data samples (ranging from 50 to 3500),

although specific details regarding data distribution among clients and the distribution of samples across all classes were not provided.

Taslimasa et al. [14] proposed ImageFed, a practical IDS designed to preserve privacy, which utilizes federated Convolutional Neural Networks. To simulate a non-IID setting, data were distributed to vehicles using a Dirichlet (μ) distribution, with (μ) values varying from 0.1 to 0.7. To assess ImageFed's resilience, they examined two potential scenarios leading to performance decline in FL: non-IID clients and limited availability of training data. Table 1 summarizes previous works and highlights our contribution. In cases where the aggregation function is not explicitly indicated, as in [9,10], we assume that FedAvg is the method employed.

Although previous works have deployed their in-vehicle IDSs in an FL environment, they exhibit some limitations. Firstly, all the proposed FL-based in-vehicle IDSs in the literature utilized a standard cloud-based FL architecture to deploy and evaluate their proposed IDSs, relying solely on one central aggregator. However, this standard FL architecture can limit the IDS's adaptability and effectiveness in detecting diverse, evolving threats. Additionally, it leads to network congestion and communication delays due to the time required to share the model between the central aggregator and a large number of vehicles, potentially millions [16]. Furthermore, relying on one aggregator poses a risk as a single point of failure [19], compromising the IDS's robustness and scalability. To address these issues, we propose an H-FL framework that incorporates multiple edge aggregators in addition to the central aggregator. This approach enables the IDS to learn from a broader range of driving scenarios and evolving threats while distributing the computational load and reducing latency. This approach aims to overcome the limitations of relying solely on a central aggregator [19].

2.3. Non-Independent and Identically Distributed Data Distributions

Non-IID data occur when the training data on each client in FL vary significantly, resulting in differing data distributions among clients [17]. In real-world applications, such data are typically non-IID because of variations in user behavior, preferences, and environments. Managing non-IID data is a major challenge in FL [20]. However, previous works [9–12] did not account for non-IID data and instead employed divisions where clients received either an equal number of samples or samples from all classes (i.e., types of attacks). This approach contradicts the nature of FL scenarios characterized by non-IID data distributions [7] and assumes an unrealistic data distribution [21]. Non-IID data distributions are only considered in [13,14]. In [13], nine candidate clients are assumed, each possessing varying numbers of data samples (50, 100, 150, 1000, 1500, 2000, 2500, 3000, 3500), but the data distribution among clients and the distribution of samples across all classes are not clarified. In [14], to achieve a non-IID setting, data are allocated to vehicles using a Dirichlet (μ) distribution, with the (μ) parameter adjusted from 0.1 to 0.7.

However, the majority of studies [9–11] did not consider the data heterogeneity (non-IID data) in CAN bus data when they distributed the data across clients, thus rendering their findings unrealistic [21]. This consideration is essential for two reasons: first, in real-world deployments, vehicles will not have uniform distributions of both attack and normal data; second, testing with varying non-IID levels enables a more comprehensive evaluation of IDS performance and robustness across diverse data distributions. To address these issues, we have adopted H-FL with a realistic non-IID data distribution. H-FL incorporates a central aggregator with multiple edge aggregators instead of having one central aggregator. Furthermore, both experimental results and theoretical analysis have shown that H-FL architecture leads to faster convergence and lower training time and energy consumption of the end devices compared with the conventional FL-based framework [18]. To the best of our knowledge, no prior work has deployed and evaluated the performance of an in-vehicle IDS in an H-FL environment to detect known and unknown attacks.

Table 1. FL-based IDSs for in-vehicle networks.

Ref.	Fed. Learning	U Attacks	Non-IID	Aggregation Function	Dataset	FL Implementation
[14]	Standard	x	✓	FedAvg	car-hacking [22]	PyTorch
[11]	Standard	x	x	FedAvg	HCRL CAN Intrusion Detection [23]	N/A
[10]	Standard	x	x	FedAvg	car-hacking [22], NAIST CAN attack dataset [24]	Keras, TensorFlow
[12]	Standard	✓	—	FedAvg, FedProx	READ [25]	N/A
[9]	Standard	x	x	FedAvg	Car Hacking: Attack & Defense Challenge 2020 [26]	Keras, TensorFlow
[13]	Standard	x	✓	FedAvg	HCRL CAN Intrusion Detection [23]	N/A
Our work	Hierarchical	✓	✓	FedAvg	car-hacking [22]	Flower

U Attacks: Unknown Attacks.

3. Materials and Methods

In this section, we outline the methodologies used in our experiments, including the experimental setup, the proposed H-FL architecture, dataset description, in-vehicle IDS, data partitioning across multiple FL clients, data preprocessing, model initialization and pre-training, local training, aggregation, and evaluation methods.

3.1. Experimental Setup

The implementation was conducted in Visual Studio Code. The experimental setup used a 64-bit Windows 11 Pro for Workstations operating system with an AMD Ryzen Threadripper PRO 5995WX processor, featuring 64 cores at 2701 MHz, 128 logical processors, and 128 GB of RAM. The construction, training, and evaluation of the deep learning model were conducted with Flower framework [27], TensorFlow (version 2.13.1), and Python (version 3.8.10). Flower is an open-source FL framework to build AI applications capable of training on data distributed across numerous devices. We selected Flower for its ability to seamlessly transition from conventional to federated machine learning frameworks, its compatibility with popular platforms such as TensorFlow and PyTorch, its support for diverse privacy requirements, and its flexibility in facilitating the implementation of novel approaches with minimal engineering effort [28].

3.2. Hierarchical Federated Learning Architecture

The H-FL architecture consists of a main cloud server, several edge servers (aggregators), and numerous clients (vehicles), as depicted in Figure 2. The cloud server is connected to the edge servers, each of which interacts with vehicles in different geographical locations. In this paper, we utilize one main server, two edges, and five clients per edge. The H-FL process begins with the cloud server sending the initial model to the edge servers. Each edge server then selects a subset of clients for training using a suitable selection approach. The selected clients receive the initial model and perform edge aggregations over several rounds to generate an edge model. Once the edge servers complete their aggregations, they send their aggregated models' parameters to the cloud server. The cloud server aggregates these edge models to produce a global model, which is then sent back to the edge servers for further aggregation. A detailed description of the H-FL process is provided in Table 2. To handle the variability in CAN bus data across different vehicle makes and models, edge servers select clients based on similarities in their CAN bus data, such as make or model, within specific geographical areas. This approach ensures that the global model is trained on relevant and similar data, thereby improving accuracy.

The main server, located at 127.0.0.1:8088, initiates the process by distributing the pre-trained model to its connected clients (edges). Subsequently, each edge, serving as a server at addresses 127.0.0.1:8080 and 127.0.0.1:8081, respectively, relays the pre-trained model to its connected clients (vehicles). Each edge server then starts the standard FL

process with the clients (vehicles) in its area, resulting in an edge model. These edge servers, now operating as clients, send their models back to the main server. The main server aggregates the received edge models into a global model, which it then redistributes to the edges. Finally, the edges, again functioning as servers, transmit the new global model to their clients (vehicles) to restart the FL cycle. Figure 3 illustrates this H-FL setup as implemented in Flower.

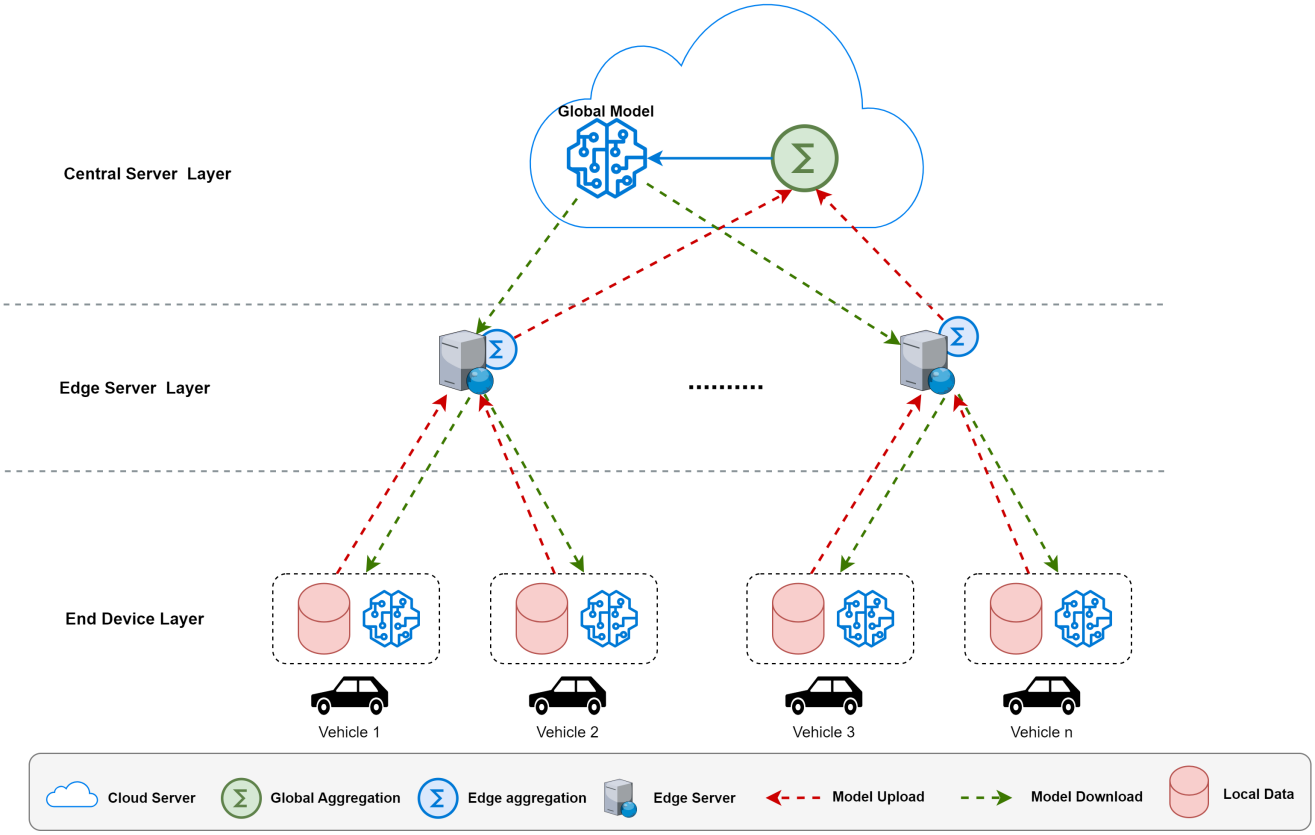


Figure 2. Architecture of the proposed H-FL method.

Table 2. Detailed description of the H-FL approach.

H-FL Process	
	Begin
Step 1	The central server $\mathcal{S}$ initializes a global model $\mathcal{GM}$
Step 2	$\mathcal{S}$ sends the $\mathcal{GM}$ to each local aggregator L_{Agg}
Step 3	Each L_{Agg} selects a subset of vehicles V_n
Step 4	L_{Agg} sends the $\mathcal{GM}$ to the selected V_n
Step 5	Each V_n trains the $\mathcal{GM}$ on local Data for K rounds
Step 6	Each V_n computes and sends the learned parameters to the corresponding L_{Agg}
Step 7	Each L_{Agg} aggregates the received local model parameters from V_n to obtain the edge aggregation model
Step 8	All L_{Agg} send the aggregated model parameters to $\mathcal{S}$ to build a new updated $\mathcal{GM}$
Step 9	The $\mathcal{S}$ aggregate edge aggregation model and build a new $\mathcal{GM}$
Step 10	Repeat steps 2–9 until achieving the desired performance
	End

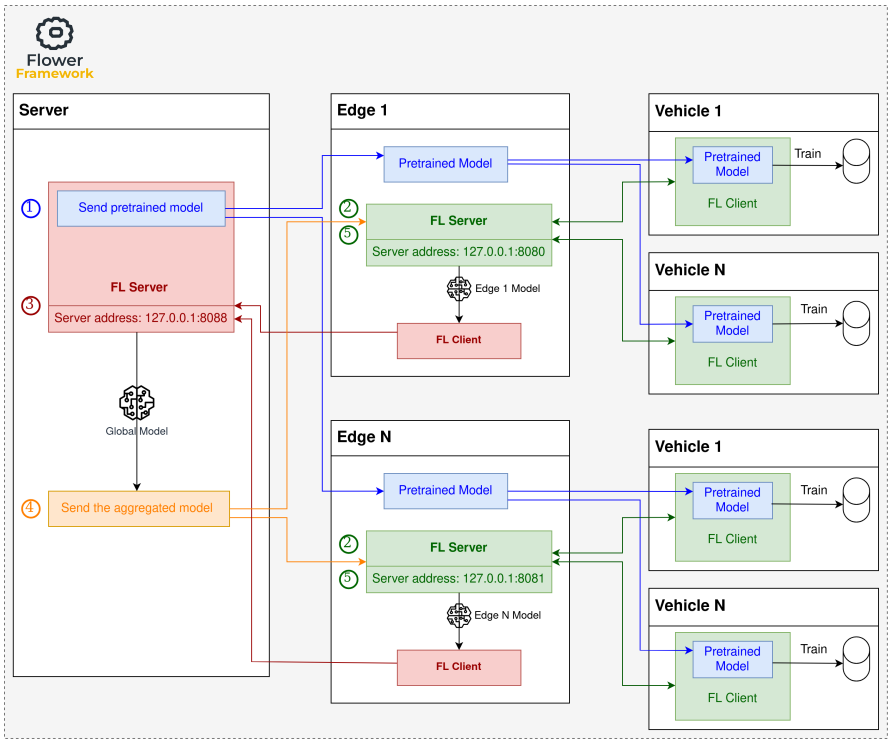


Figure 3. H-FL environment in Flower.

3.3. Dataset Description

To evaluate the performance of our IDS within the federated framework, we employed a benchmark car hacking dataset published by Song et al. [29]. We selected this dataset due to its widespread use in automotive security research and its status as the most frequently cited resource in the literature of the field [30]. The dataset contains normal data and four types of attacks: Denial of Service (DoS), frame fuzzification, engine RPM spoofing, and drive gear spoofing. It includes four files, one for each attack (DoS, frame fuzzification, gear spoofing, and RPM spoofing), with each file including normal as well as attack instances. Table 3 shows the number of instances for each type of attack as well as for normal data. For every CAN message, the dataset provides valuable information, including timestamp, CAN ID, DLC, data field, and flag. The timestamp records the exact time the message was captured since the system startup, while the CAN ID determines message priority, with lower values being prioritized. Additionally, the DLC defines the length of the data field, which can reach a maximum of 8 bytes. The flag indicates whether the message is normal or an attack. Table 4 provides an overview of the dataset's features, along with their descriptions and data types. To ensure the generalizability of our proposed H-FL architecture across diverse scenarios and to avoid reliance on a single dataset, we validated it using another benchmark dataset in automotive security research, the car hacking: attack & defense challenge 2020 dataset [26], which comprises normal traffic and three types of attacks: DoS, frame fuzzification, and spoofing [26].

Table 3. Dataset overview.

Attack Type	Attack Instances	Normal Instances
DoS	587,521	3,078,250
Frame Fuzzification	491,847	3,347,013
Gear	597,252	3,845,890
RPM	654,897	3,966,805
Total	2,331,517	14,237,958

Table 4. Data features, descriptions, and types.

Feature	Describtion	Type
Timestamp	Time	float
CAN ID	CAN message identifier	hexadecimal
DLC	The length of the data field, measured in bytes	integer
Data	Payload (64-bit)	hexadecimal
Flag	T or R, T: Attack, R: Normal	string

3.4. In-Vehicle Intrusion Detection System

In this paper, we evaluate the performance of our proposed IDS as detailed in [15]. The proposed IDS consists of a multistage approach, where the first stage is responsible for detecting and classifying known attacks using an Artificial Neural Network (ANN), while the second stage acts as an anomaly detector to identify novel, unknown attacks using a Long Short-Term Memory (LSTM) autoencoder. The first stage of our proposed in-vehicle IDS classifies traffic data as either normal traffic or any known attack class that the ANN model has been trained on. Any data classified as normal by the first model will be re-examined by the second model (LSTM autoencoder) to detect unseen attacks that bypassed the initial model, providing an additional layer of protection. The first stage is a signature-based multiclassifier model, while the second stage is an unsupervised anomaly detection model. Figure 4 shows the workflow of proposed multistage IDS. The proposed IDS achieves an F1-score of 0.95 and a 99.99% detection rate for previously unseen attacks, all within a compact model size of 2.98 MB, making it highly suitable for deployment in real-world applications. This efficient model size fits well within the memory constraints of vehicle-level systems, which typically have over 1 GB of available RAM [31]. More details about the proposed IDS can be found in [15].

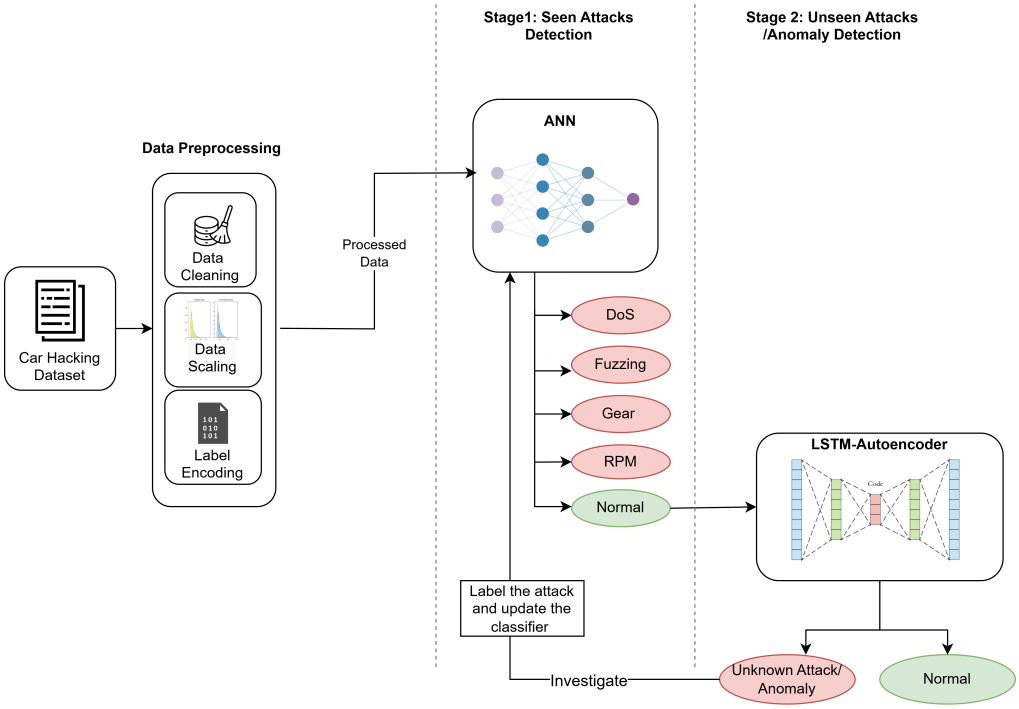


Figure 4. Workflow of the proposed multistage in-vehicle IDS.

3.5. Data Preprocessing and Partitioning

To replicate a real-world scenario where data are independently generated for each client, each FL client performs individual and independent data preprocessing before engaging in local model training, as depicted in Figure 5. However, certain preprocessing steps, such as removing missing data and label encoding, should occur before data partitioning. This approach ensures that each client receives a consistent and complete dataset, fostering a standardized training environment and maintaining uniform label representation throughout the FL process. Therefore, we divide the data preprocessing into two phases: initial data preprocessing and on-device data preprocessing. Initial data preprocessing occurs before data partitioning, while on-device preprocessing is performed on the device. This section will elaborate on each step: initial data preprocessing, data partitioning, and on-device data preprocessing.

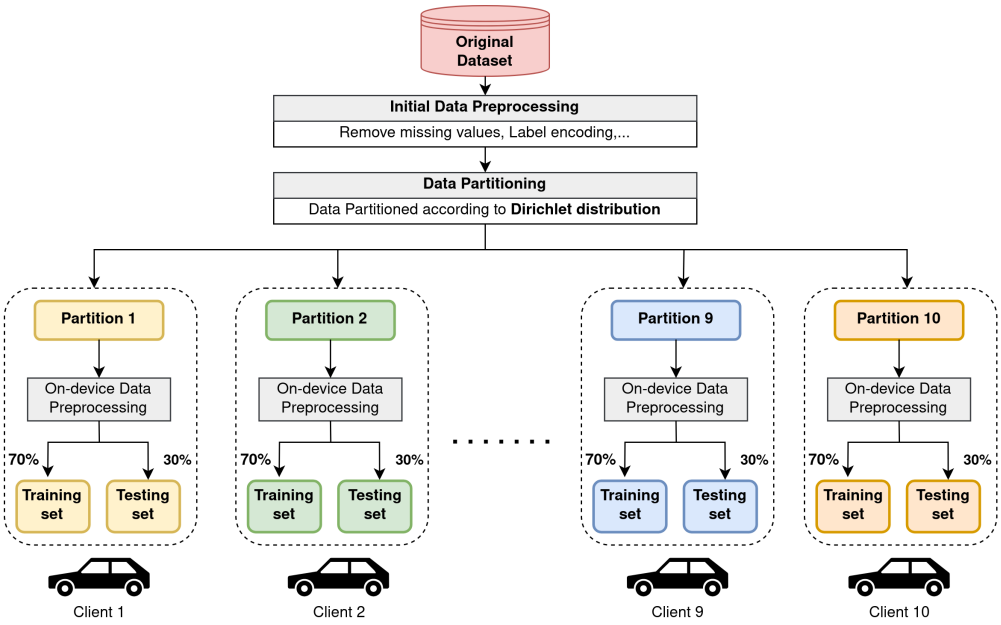


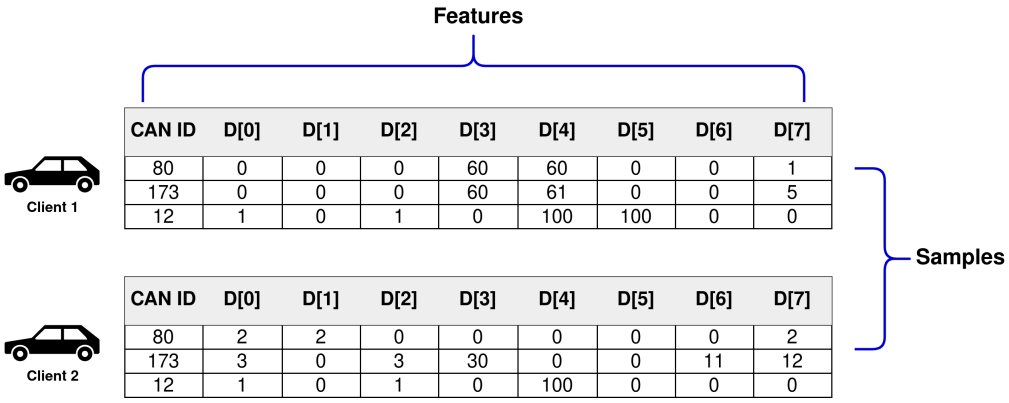
Figure 5. Data partitioning.

3.5.1. Initial Data Preprocessing

In the initial phase of data preprocessing, for each data file, we convert 'T' to the corresponding attack type, such as 'DoS', and 'R' to 'Normal' in the 'Flag' column. Then, we shift the 'Flag' field to the last column and fill non-available data bytes with (NaN) values. We combine the four data files into one. Given the extensive data points we have, we remove any row with these missing values (NAN) in the Data fields. Lastly, we use a label encoder to convert categorical values into numerical representations. This ensures consistent encoding across all clients, facilitating model training and aggregation.

3.5.2. Data Partitioning

After the initial data preprocessing step, we perform data partitioning to simulate the FL environment. In the context of CAN bus data, the data are horizontally structured, where the training data from participating clients shares a common feature space but exhibits distinct sample spaces. As illustrated in Figure 6, for instance, client 1 and client 2 contain dissimilar data while sharing the same set of features.

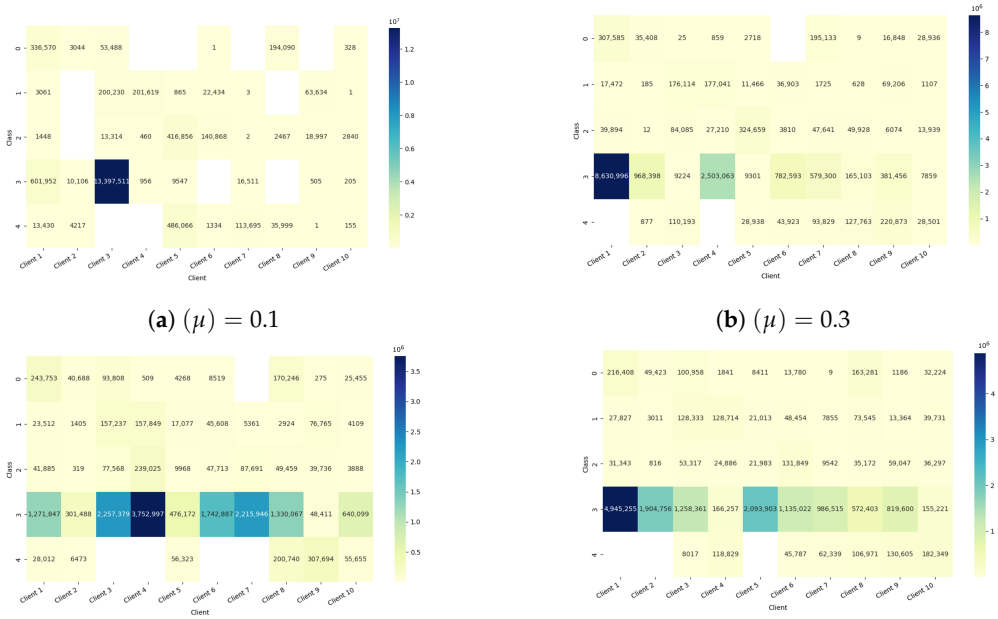


CAN ID	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]
80	0	0	0	60	60	0	0	1
173	0	0	0	60	61	0	0	5
12	1	0	1	0	100	100	0	0

CAN ID	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]
80	2	2	0	0	0	0	0	2
173	3	0	3	30	0	0	11	12
12	1	0	1	0	100	0	0	0

Figure 6. Horizontal CAN bus data.

To simulate a non-IID setting, we assign data to clients using Dirichlet $(Dir(\mu))$ distribution according to [20], where every client is assigned a share of the samples for each label. Dirichlet distribution is commonly used as a prior distribution in Bayesian statistics [32] and is an appropriate choice to simulate real-world data distribution [20]. An advantage of this approach is that we can flexibly change the imbalance level by varying the concentration parameter (μ). The parameter (μ) in $(Dir(\mu))$ regulates the degree of non-IID property among clients, where lower (μ) values signify a higher level of non-IID. We adjust (μ) to simulate diverse data distributions among clients, ranging from 0.1 (indicating the highest non-IID) to 0.7 (indicating the lowest non-IID). Figure 7 shows the data partition on car hacking dataset with different (μ) values. The number displayed in each rectangle represents the quantity of data samples of a class (DoS (0), frame fuzzification (1), drive gear spoofing (2), normal (3), and engine RPM spoofing (4)) for each client. However, we exclude $(\mu) = 0.1$ due to its highly imbalanced data distribution, as shown in Figure 7a, where some clients lack normal data (3) for training, making it unrealistic. Data are distributed among two edges, each with five clients, for a total of ten clients. We chose ten clients, a common practice in the literature [12,33–35], to manage the increased computational demands associated with additional clients.



(a) $(\mu) = 0.1$ **(b)** $(\mu) = 0.3$

(c) $(\mu) = 0.5$ **(d)** $(\mu) = 0.7$

Figure 7. Dataset distribution based on Dirichlet(μ) distribution.

3.5.3. On-Device Data Preprocessing

For on-device data preprocessing, unwanted columns (Timestamp, DLC) are removed. Additionally, we convert the CAN ID and Data values from hexadecimal to decimal values per the machine learning requirement, and a normalization process is applied.

3.6. Model Initialization and Pre-Training

The model is initialized on the server side, followed by distributing a copy to each participating client. In this paper, we use a pre-trained model, as Chen et al. [36] demonstrates that pre-training models on a central server before distribution to edge devices enhance task accuracy, especially with non-IID client data. Additionally, many advantages arise from pre-training a model on a centralized dataset for FL. Firstly, it enhances generalization by enabling the model to capture general features and patterns, ensuring better adaptability to the diverse data distributions across devices. Furthermore, initializing a pre-trained model on a powerful centralized server can significantly reduce the time required to fine-tune the model for specific devices [37]. To pre-train the models, we extract a small data sample representing each class, amounting to 0.8% (less than 1%) of the entire original dataset, for use in pre-training. The remaining data are then partitioned across clients. Our approach builds on the methodology outlined in [31], employing K-Means clustering to capture underlying patterns and the Synthetic Minority Over-sampling Technique (SMOTE) to address class imbalance. The sampling process starts by grouping the data into clusters using K-Means. To ensure a representative subset, we apply a group-based sampling strategy, selecting 0.8% of data points from each cluster. This approach retains diversity in the sampled dataset while significantly reducing its size. The selected samples are subsequently excluded from the full dataset before partitioning the data among clients. As the sampled data often display label imbalance, SMOTE is applied to effectively balance the classes.

3.7. Local Training

On the vehicle side, we include all clients in the training and evaluation process to ensure consistency in the results. Clients receive the pre-trained model from the corresponding edge and train it on their private data locally for one epoch and fifty rounds. After each round, each client sends its local weight updates to the corresponding edge for aggregation.

3.8. Aggregation

In H-FL, there are two stages of aggregation: edge aggregation and server aggregation. Edge aggregation combines the models from the clients, while server aggregation merges the results from edge aggregation to produce a global model. An aggregation strategy refers to the process of combining local models from clients into a centralized global model. In this paper, we use for both edge server aggregation, the standard aggregation strategy, and federated averaging (FedAvg) [17] according to the following equation:

$$w_{t+1}^{g} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^{k} \tag{1}$$

Accordingly, the edge server calculates the weighted average of local w_k^{t+1} based on the number of samples each local vehicle used in one round (n_k) over the total training samples (n). Similarly, the cloud server calculates the weighted average of the total training samples (n) under each edge server.

3.9. Evaluation

In FL, there are two ways to evaluate the model: on the server side or the client side. In this paper, we chose to employ client-side evaluation over server-side evaluation because it enables us to assess models across a larger set of data, often leading to more realistic evaluation outcomes [38].

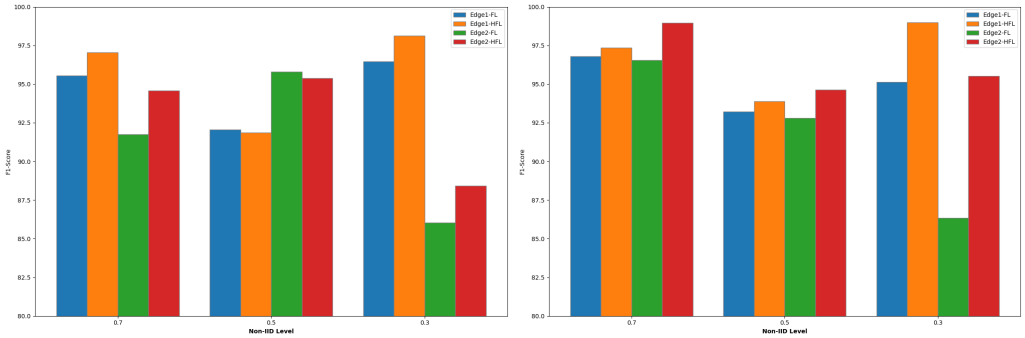
4. Results and Evaluations

4.1. Evaluation Metrics

For a robust assessment of model performance, it is important to choose metrics that align with the dataset type and class distribution. In the context of imbalanced intrusion datasets, relying solely on accuracy is considered unreliable [39]. Hence, we opted for the evaluation metrics of F1-score, precision, and recall for our model. Furthermore, given the unequal distribution of the data based on the label, we use the weighted average to consider the size of each class. This is an important consideration, as it ensures that the distribution of instances in each class is taken into account [40]. Additionally, the training time for each model, network load, and memory requirements for each client are also evaluated.

4.2. Performance Results and Analysis

We evaluate the performance of the proposed IDS within an H-FL architecture using two datasets, each with three non-IIDlevels, resulting in a total of 24 scenarios: 12 for dataset 1 and 12 for dataset 2. Figure 8 shows the average F1-score performance of the ANN model and the proposed cascaded IDS across different non-IID levels in the car hacking dataset [29]. As depicted in Figure 8a, the ANN classifier's performance improved at all non-IID levels except when $(\mu = 0.5)$, where it slightly decreased after applying the H-FL model. In contrast, the F1-score for the cascaded multistage IDS increased across all non-IID levels, as shown in Figure 8b. Notably, the H-FL improved the F1-score in 10 out of 12 scenarios in the car hacking dataset.



(**a**) ANN model (**b**) Proposed multistage-IDS

Figure 8. Average F1-score results non-IID levels across communication rounds for car hacking dataset [29].

This indicates that deploying the model in an H-FL architecture significantly enhanced performance, enabling it to more effectively identify both known and unknown attacks by not only learning from participating clients' data but also from a broader range of client data and evolving threats. The reasoning behind this is that an edge-FL approach, when applied to a limited number of vehicles with specific data and attack scenarios, confines the learning process to the vehicles participating within that particular edge-FL network. This restriction can limit the model's ability to generalize to diverse attack patterns not represented in the local FL process. In contrast, H-FL aggregates models from multiple edge servers across different areas into a central server, then redistributes the global model back to the edges and ultimately to all participating vehicles. This process enables vehicles to leverage insights from a broader range of data and attack scenarios across various regions, enhancing the IDS's adaptability to new and varied threats, and ultimately increasing robustness and accuracy in attack detection across the network.

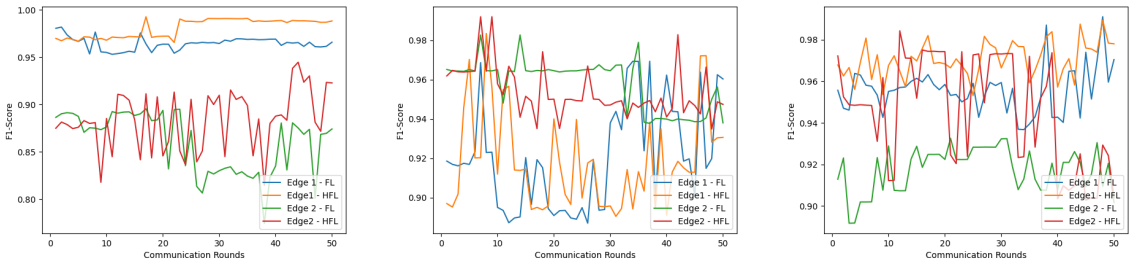
Figure 8b also reveals a notable difference between Edge 1 and Edge 2, primarily due to the high imbalance in data distribution when $(\mu = 0.3)$, compared to $(\mu = 0.5)$ and $(\mu = 0.7)$. Across all non-IID distributions, the H-FL approach consistently outperforms the edge-FL approach, with percentage increases ranging from 0.57% to 10.63%, highlighting a significant performance improvement with the H-FL method. The increase may vary based

on the clients selected under each edge and non-IID level, as demonstrated by the Edge 2 results in Figure 8b. These results underscore the critical impact of the participating clients' data on performance. Detailed F1-score performance is available in Figures 9 and 10, which further illustrate the F1-score performance of the ANN model and the proposed cascaded IDS across various non-IID levels. Table 5 shows the average recall and precision of our IDS in an H-FL environment. The results indicate an increase in both recall and precision after deploying the H-FL model at nearly all non-IID levels. The best results are obtained when the data are distributed across clients with a low level of imbalance, such as at $(\mu = 0.7)$. Figure 11 demonstrates the distributed loss across communication rounds. Results show that the model has different convergence speeds across non-IID levels.

Table 5. Average recall and precision of the proposed IDS model.

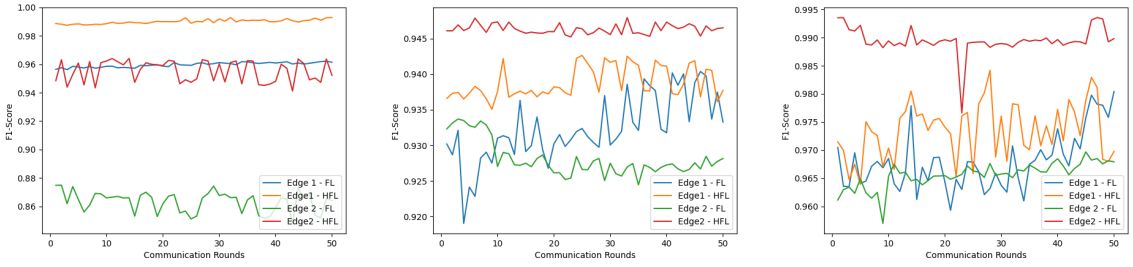
Non-IID	Edge 1 FL		Edge 1 H-FL		Edge 2 FL		Edge 2 H-FL	
	Recall	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**
0.3	94.93	96.76	98.39	99.89	81.29	84.61	86.48	94.71
0.5	92.59	95.27	93.91	94.21	92.92	93.11	95.40	94.16
0.7	95.28	98.63	95.75	99.53	96.27	97.09	98.14	99.86

We also evaluated the performance of the proposed IDS using another dataset, the car hacking: attack & defense challenge 2020 dataset. Figure 12 shows that 50% of the scenarios demonstrated improvement, while the remaining 50% showed a decline. This is likely due to imbalanced data distributions between labels and data heterogeneity. Overall, H-FL improved the F1-score in 16 out of 24 evaluated scenarios across both datasets.



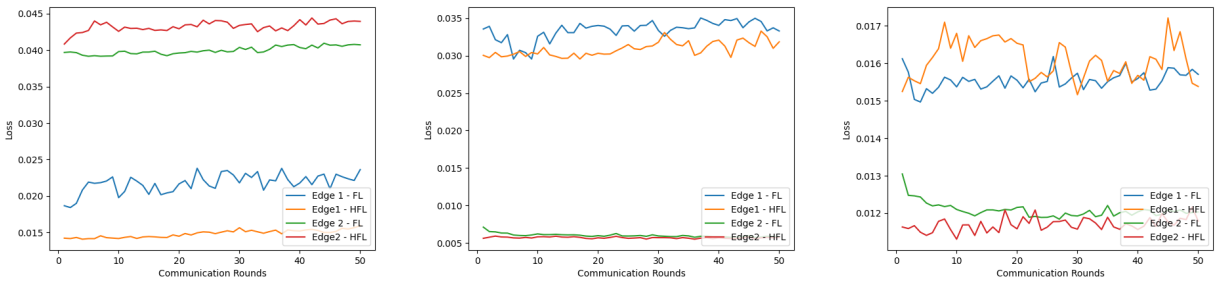
(**a**) $(\mu) = 0.3$ (**b**) $(\mu) = 0.5$ (**c**) $(\mu) = 0.7$

Figure 9. F1-score of ANN model for different non-IID levels across communication rounds.



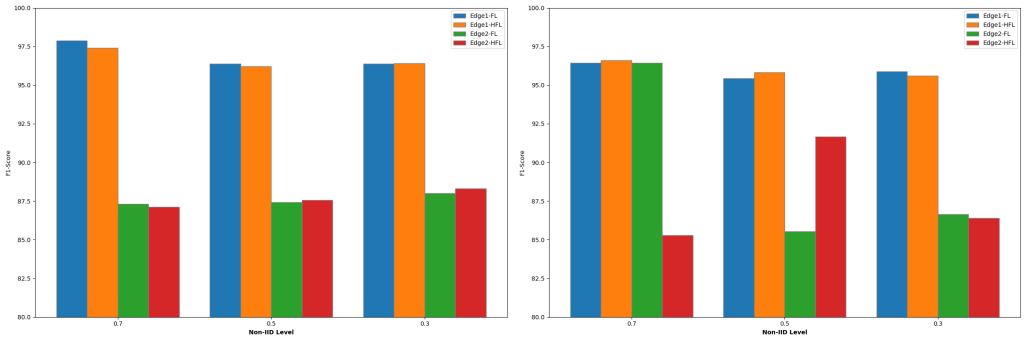
(**a**) $(\mu) = 0.3$ (**b**) $(\mu) = 0.5$ (**c**) $(\mu) = 0.7$

Figure 10. F1-score of our proposed multistage-IDS for different non-IID levels across communication rounds.



(**a**) $(\mu) = 0.3$ (**b**) $(\mu) = 0.5$ (**c**) $(\mu) = 0.7$

Figure 11. Distributed loss across communication rounds.



(**a**) ANN model (**b**) Proposed multistage-IDS

Figure 12. Average F1-score results non-IID levels across communication rounds for car hacking: Attack & Defense Challenge 2020 dataset [26].

Table 6 shows the average training time in seconds (S) for both the ANN and LSTM models, as well as the total training time for one round. The results indicate that training time depends on the number of training data, with Client 6 having the highest training time of 75.35 s due to its large training dataset, and the inverse being true for clients with smaller datasets.

Table 6. Average training time per client for one round.

Clients	Number of Training Data	Average ANN Training Time (S)	Average LSTM Training Time (S)	Total Training Time (S)
Client 0	75,341	1.45	4.49	5.94
Client 1	956,504	14.47	56.02	70.49
Client 2	140,115	2.53	10.61	13.14
Client 3	440,494	7.25	27.03	34.28
Client 4	410,876	6.88	27.09	33.97
Client 5	140,044	2.58	0.41	2.99
Client 6	1,118,686	16.71	58.64	75.35
Client 7	405,376	6.34	17.77	24.11
Client 8	131,100	2.39	7.55	9.94
Client 9	162,991	2.93	2.28	5.21

We also measured the network load by calculating the size of weights transferred between the server and edges, as well as between vehicles and edges. The weights size is 0.966 MB. As shown in Table 7, the network load for a single client in one round is twice the weights size, as each client sends its weights to the edge and receives the newly aggregated weights. In H-FL, the network load slightly increases because the main server sends the global model weights to the edges and receives the weights from each edge for aggregation. Although H-FL introduces an additional layer of communication between the server and

edges, which is absent in standard FL, it reduces bottlenecks by distributing clients across edges, effectively dividing the network load and improving efficiency.

Table 7. Network load across different architectures.

Architecture	Edges	Clients	Rounds	Model Weights (MB)
Edge-FL/FL	1	1	1	1.932
H-FL	1	1	1	3.86
Edge-FL /FL	1	5	50	483.45
Edge- H-FL	1	5	50	968.8
FL	1	10	50	966.9
H-FL	2	10	50	970.76

One of the main challenges in implementing FL is the memory consumption of learning models trained locally on individual vehicles. High-end automobiles, such as those produced by Tesla, utilize advanced deep learning algorithms for tasks like object detection and various driver assistance features. To handle the substantial computational demands of data from cameras and sensors, these vehicles are equipped with dedicated GPUs. For example, Tesla's latest Model S and Model X are powered by AMD RDNA 2 GPUs, which provide 16–32 GB of memory for processing tasks [41]. Given that our IDS also relies on GPUs, we evaluate a vehicle's memory requirements by analyzing both the size of the learning models and the memory needed to process the dataset. The total model size is 2.98 MB. To optimize memory usage during training, we process the data in batches of 256 rather than loading the entire dataset at once. This approach significantly reduces the memory required for training data from 104.25 MB (for the full dataset) to just 0.02 MB per batch. Consequently, the model requires approximately 1 GB of memory for operation, which is well within the capabilities of current vehicle-level machines.

5. Conclusions and Future Works

The aim of this paper was to deploy and evaluate our proposed in-vehicle IDS [15] within the H-FL framework to address the limitations of the standard FL approach. We evaluated the performance of the IDS using two datasets with three non-IID levels to simulate real-world scenarios and assess the impact of different data distributions. Experimental results indicate that the H-FL framework, in most scenarios, improves F1-score results by up to 10.63% for both known and unknown attacks across diverse non-IID conditions, outperforming edge-FL, which is constrained by smaller, vehicle-specific datasets. However, in certain scenarios, the results showed a decline after applying H-FL, potentially due to the heterogeneity of data among the clients. Overall, H-FL improved the F1-score in 16 out of 24 evaluated scenarios in two datasets. This raises an important research question for future studies: How can we cluster vehicles to ensure the effectiveness of the H-FL framework? By applying clustering methods, vehicles could be grouped based on factors such as proximity, connectivity, or data similarity, which could improve overall training performance across all clients [42]. One limitation of this work is the use of a fixed number of clients for all experiments, whereas in real-world scenarios, the number of clients can vary significantly. Future work should include a more flexible evaluation framework that tests variations in client numbers to enhance the reliability and scalability of the proposed H-FL architecture.

Author Contributions: Conceptualization, M.A., A.J., O.R. and T.S.; Methodology, M.A.; Supervision, A.J. and O.R.; Writing—original draft, M.A.; Writing—review and editing, M.A. and A.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets used in this paper are publicly available to everyone and can be obtained from https://ocslab.hksecurity.net/Datasets/car-hacking-dataset and

https://ocslab.hksecurity.net/Datasets/carchallenge2020 (accessed on 1 November 2022) and has been cited in the manuscript.

Acknowledgments: We would like to acknowledge the scholarship and support provided by Majmaah University.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Pickford, J.; Attale, R.; Shaikh, S.; Nguyen, H.N.; Harrison, L. Systematic Risk Characterisation of Hardware Threats to Automotive Systems. *J. Auton. Transp. Syst.* **2024**, *1*, 1–36. [CrossRef]
2. Grand View Research, Inc. Connected Car Market Size & Share Analysis Report. 2030. Available online: https://www.grandviewresearch.com/industry-analysis/connected-car-market (accessed on 7 August 2024).
3. Paul, A.; Islam, M.R. An artificial neural network based anomaly detection method in can bus messages in vehicles. In Proceedings of the 2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI), Rajshahi, Bangladesh, 8–9 July 2021; pp. 1–5. [CrossRef]
4. Young, C.; Zambreno, J.; Olufowobi, H.; Bloom, G. Survey of automotive controller area network intrusion detection systems. *IEEE Des. Test* **2019**, *36*, 48–55. [CrossRef]
5. Alfardus, A.; Rawat, D.B. Intrusion detection system for can bus in-vehicle network based on machine learning algorithms. In Proceedings of the 2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 1–4 December 2021; pp. 944–949. [CrossRef]
6. Alsamiri, J.; Alsubhi, K. Federated Learning for Intrusion Detection Systems in Internet of Vehicles: A General Taxonomy, Applications, and Future Directions. *Future Internet* **2023**, *15*, 403. [CrossRef]
7. Hernandez-Ramos, J.L.; Karopoulos, G.; Chatzoglou, E.; Kouliaridis, V.; Marmol, E.; Gonzalez-Vidal, A.; Kambourakis, G. Intrusion Detection based on Federated Learning: A systematic review. *arXiv* **2023**, arXiv:2308.09522. [CrossRef]
8. Group, X.W. *Guidelines for an Intrusion Detection System for In-Vehicle Networks*; Technical Report X.1375; International Telecommunication Union: Geneva, Switzerland, 2020.
9. Driss, M.; Almomani, I.; e Huma, Z.; Ahmad, J. A federated learning framework for cyberattack detection in vehicular sensor networks. *Complex Intell. Syst.* **2022**, *8*, 4221–4235. [CrossRef]
10. Shibly, K.H.; Hossain, M.D.; Inoue, H.; Taenaka, Y.; Kadobayashi, Y. Personalized federated learning for automotive intrusion detection systems. In Proceedings of the 2022 IEEE Future Networks World Forum (FNWF), Montreal, QC, Canada, 10–14 October 2022; pp. 544–549. [CrossRef]
11. Yu, T.; Hua, G.; Wang, H.; Yang, J.; Hu, J. Federated-LSTM based Network Intrusion Detection Method for Intelligent Connected Vehicles. In Proceedings of the IEEE International Conference on Communications (ICC), Seoul, Republic of Korea, 16–20 May 2022. [CrossRef]
12. Zhang, H.; Zeng, K.; Lin, S. Federated graph neural network for fast anomaly detection in controller area networks. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 1566–1579. [CrossRef]
13. Yang, J.; Hu, J.; Yu, T. Federated AI-Enabled In-Vehicle Network Intrusion Detection for Internet of Vehicles. *Electronics* **2022**, *11*, 3658. [CrossRef]
14. Taslimasa, H.; Dadkhah, S.; Neto, E.P.; Xiong, P.; Iqbal, S.; Ray, S.; Ghorbani, A. ImageFed: Practical Privacy Preserving Intrusion Detection System for In-Vehicle CAN Bus Protocol. In Proceedings of the 2023 IEEE 9th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), New York, NY, USA, 6–8 May 2023. [CrossRef]
15. Althunayyan, M.; Javed, A.; Rana, O. A robust multi-stage intrusion detection system for in-vehicle network security using hierarchical federated learning. *Veh. Commun.* **2024**, *49*, 100837. [CrossRef]
16. Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečnỳ, J.; Mazzocchi, S.; McMahan, B.; et al. Towards federated learning at scale: System design. *Proc. Mach. Learn. Syst.* **2019**, *1*, 374–388. [CrossRef]
17. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282. [CrossRef]
18. Liu, L.; Zhang, J.; Song, S.; Letaief, K.B. Client-edge-cloud hierarchical federated learning. In Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6. [CrossRef]
19. Rana, O.; Spyridopoulos, T.; Hudson, N.; Baughman, M.; Chard, K.; Foster, I.; Khan, A. Hierarchical and Decentralised Federated Learning. *arXiv* **2023**, arXiv:2304.14982. [CrossRef]
20. Li, Q.; Diao, Y.; Chen, Q.; He, B. Federated learning on non-iid data silos: An experimental study. In Proceedings of the 2022 IEEE 38th International Conference on Data Engineering (ICDE), Virtual, 9–12 May 2022; pp. 965–978. [CrossRef]
21. Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; Chandra, V. Federated learning with non-iid data. *arXiv* **2018**, arXiv:1806.00582. [CrossRef]
22. Song, H.M.; Woo, J.; Kim, H.K. In-vehicle network intrusion detection using deep convolutional neural network. *Veh. Commun.* **2020**, *21*, 100198. [CrossRef]

23. Lee, H.; Jeong, S.H.; Kim, H.K. OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame. In Proceedings of the 2017 15th Annual Conference on Privacy, Security and Trust (PST), Calgary, AB, Canada, 28–30 August 2017; pp. 57–5709. [CrossRef]
24. Hossain, M.D.; Inoue, H.; Ochiai, H.; Fall, D.; Kadobayashi, Y. LSTM-based intrusion detection system for in-vehicle can bus communications. *IEEE Access* **2020**, *8*, 185489–185502. [CrossRef]
25. Marchetti, M.; Stabili, D. READ: Reverse engineering of automotive data frames. *IEEE Trans. Inf. Forensics Secur.* **2018**, *14*, 1083–1097. [CrossRef]
26. Kang, H.; Kwak, B.I.; Lee, Y.H.; Lee, H.; Lee, H.; Kim, H.K. Car hacking and defense competition on in-vehicle network. In Proceedings of the Workshop on Automotive and Autonomous Vehicle Security (AutoSec), online, 25 February 2021; Volume 2021, p. 25. [CrossRef]
27. AI, F. Flower A Friendly Federated Learning Framework. 2024. Available online: https://flower.ai/ (accessed on 10 January 2024).
28. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Li, K.H.; Parcollet, T.; de Gusmão, P.P.B.; et al. Flower: A friendly federated learning research framework. *arXiv* **2020**, arXiv:2007.14390. [CrossRef]
29. Seo, E.; Song, H.M.; Kim, H.K. GIDS: GAN based intrusion detection system for in-vehicle network. In Proceedings of the 2018 16th Annual Conference on Privacy, Security and Trust (PST), Belfast, Ireland, 28–30 August 2018; pp. 1–6. [CrossRef]
30. Rajapaksha, S.; Kalutarage, H.; Al-Kadri, M.O.; Petrovski, A.; Madzudzo, G.; Cheah, M. Ai-based intrusion detection systems for in-vehicle networks: A survey. *ACM Comput. Surv.* **2023**, *55*, 1–40. [CrossRef]
31. Yang, L.; Moubayed, A.; Shami, A. MTH-IDS: A Multitiered Hybrid Intrusion Detection System for Internet of Vehicles. *IEEE Internet Things J.* **2022**, *9*, 616–632. [CrossRef]
32. Huang, J. Maximum likelihood estimation of Dirichlet distribution parameters. *CMU Tech. Rep.* **2005**, *18*, pp. 1–9.
33. Caldas, S.; Konečny, J.; McMahan, H.B.; Talwalkar, A. Expanding the reach of federated learning by reducing client resource requirements. *arXiv* **2018**, arXiv:1812.07210. [CrossRef]
34. Díaz, J.S.P.; García, Á.L. Study of the performance and scalability of federated learning for medical imaging with intermittent clients. *Neurocomputing* **2023**, *518*, 142–154. [CrossRef]
35. Tam, K.; Li, L.; Han, B.; Xu, C.; Fu, H. Federated noisy client learning. In *IEEE Transactions on Neural Networks and Learning Systems*; IEEE: New York, NY, USA, 2023. [CrossRef]
36. Chen, H.Y.; Tu, C.H.; Li, Z.; Shen, H.W.; Chao, W.L. On the importance and applicability of pre-training for federated learning. *arXiv* **2022**, arXiv:2206.11488. [CrossRef]
37. Kaur, I.; Jadhav, A.J. A Comprehensive Study on Model Initialization Techniques Ensuring Efficient Federated Learning. *arXiv* **2023**, arXiv:2311.02100. [CrossRef]
38. Framework, F. Use a Federated Learning Strategy. Available online: https://flower.ai/docs/framework/tutorial-series-use-a-federated-learning-strategy-pytorch.html (accessed on 24 March 2024).
39. Galar, M.; Fernandez, A.; Barrenechea, E.; Bustince, H.; Herrera, F. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2011**, *42*, 463–484. [CrossRef]
40. Belarbi, O.; Spyridopoulos, T.; Anthi, E.; Mavromatis, I.; Carnelli, P.; Khan, A. Federated deep learning for intrusion detection in IoT networks. In Proceedings of the GLOBECOM 2023-2023 IEEE Global Communications Conference, Kuala Lumpur, Malaysia, 4–8 December 2023; pp. 237–242. [CrossRef]
41. Lambert, F. Tesla Is Using New Amd RDNA 2 Gpu in New Model S/X, Same as in Playstation 5. 2021. Available online: https://electrek.co/2021/06/01/tesla-amd-rdna-2-gpu-new-model-s-x-same-playstation-5/ (accessed on 18 November 2024).
42. Briggs, C.; Fan, Z.; Andras, P. Federated learning with hierarchical clustering of local updates to improve training on non-IID data. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–9.