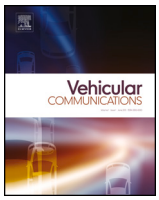
Vehicular Communications 49 (2024) 100837

Contents lists available at ScienceDirect

Vehicular Communications

journal homepage: www.elsevier.com/locate/vehcom



A robust multi-stage intrusion detection system for in-vehicle network security using hierarchical federated learning

Muzun Althunayyan [a,b,*], Amir Javed [a], Omer Rana [a]

[a] Cardiff University, School of Computer Science & Informatics, Cardiff, CF24 3AA, United Kingdom
[b] Majmaah University, Computer Sciences and Information Technology College, Al Majma'ah, 15362, Saudi Arabia

ARTICLE INFO

Keywords:
CAN bus
Cyberattack
Intrusion detection system
Anomaly detection
In-vehicle network

ABSTRACT

As connected and autonomous vehicles proliferate, the Controller Area Network (CAN) bus has become the predominant communication standard for in-vehicle networks due to its speed and efficiency. However, the CAN bus lacks basic security measures such as authentication and encryption, making it highly vulnerable to cyberattacks. To ensure in-vehicle security, intrusion detection systems (IDSs) must detect seen attacks and provide a robust defense against new, unseen attacks while remaining lightweight for practical deployment. Previous work has relied solely on the CAN ID feature or has used traditional machine learning (ML) approaches with manual feature extraction. These approaches overlook other exploitable features, making it challenging to adapt to new unseen attack variants and compromising security. This paper introduces a cutting-edge, novel, lightweight, in-vehicle, IDS-leveraging, deep learning (DL) algorithm to address these limitations. The proposed IDS employs a multi-stage approach: an artificial neural network (ANN) in the first stage to detect seen attacks, and a Long Short-Term Memory (LSTM) autoencoder in the second stage to detect new, unseen attacks. To understand and analyze diverse driving behaviors, update the model with the latest attack patterns, and preserve data privacy, we propose a theoretical framework to deploy our IDS in a hierarchical federated learning (H-FL) environment. Experimental results demonstrate that our IDS achieves an F1-score exceeding 0.99 for seen attacks and exceeding 0.95 for novel attacks, with a detection rate of 99.99%. Additionally, the false alarm rate (FAR) is exceptionally low at 0.016%, minimizing false alarms. Despite using DL algorithms known for their effectiveness in identifying sophisticated and zero-day attacks, the IDS remains lightweight, ensuring its feasibility for real-world deployment. This makes our model robust against seen and unseen attacks.

1. Introduction

In connected and autonomous vehicles (CAVs), many mechanical components have been replaced by electronic components [1]. These vehicles contain numerous Electronic Control Units (ECUs) connected via various standard automobile in-vehicle communication protocols, such as Controller Area Network (CAN), FlexRay, Local Interconnect Network (LIN), and Media Oriented System Transport (MOST). Among these protocols, CAN is considered the de facto protocol for in-vehicle communication [2] due to its features including high speed, noise cancellation, and ease of use. Although it was initially developed for industrial machines, it has since been adopted for in-vehicle network communications. However, it lacks basic security features such as sender authentication and encryption [3]. The main reasons for not implementing these security measures in in-vehicle networks are the intensive use of the vehicle's limited computational resources and the resultant increased latency, which could potentially lead to a failure to meet critical safety-related deadlines [4,5]. Thus, any security measure designed should be lightweight to ensure ease of deployment.

Furthermore, interconnectivity in modern vehicles introduces attack surfaces that expose the vehicle to cyberattacks. The attack surfaces can be accessed physically or remotely [6]. Physical access can be made via USB, CD player, onboard diagnostic (OBD)-II port, and so on. In addition, remote access can be made via short-range wireless technologies such as Bluetooth and radio frequency identification (RFID) and long-range wireless technologies such as Wi-Fi and long-term evolution (LTE). Therefore, the system is vulnerable to various cyberattacks, potentially resulting in severe consequences, including the loss of human

* Corresponding author.
E-mail addresses: AlthunayyanMS@cardiff.ac.uk (M. Althunayyan), javeda7@cardiff.ac.uk (A. Javed), ranaof@cardiff.ac.uk (O. Rana).

https://doi.org/10.1016/j.vehcom.2024.100837
Received 4 October 2023; Received in revised form 15 July 2024; Accepted 26 August 2024
Available online 30 August 2024

life [7]. There may be adverse consequences if an intruder manages to infiltrate the CAN bus system and introduce malicious messages. For instance, an unauthorized individual with access to the in-vehicle network can tamper with vital functionalities like braking, door locking mechanisms, and steering, thereby presenting a notable safety hazard. An example of a cybersecurity breach in the automotive industry can be found in [8], where two hackers were able to exploit vulnerabilities in a Jeep's system and remotely control it to perform dangerous maneuvers, including abruptly turning the steering wheel and suddenly applying the parking brake at high speeds, leading to catastrophic accidents. In the same way, hackers were able to exploit a vulnerability in the infotainment system of a General Motors car, allowing them to gain unauthorized access and steal data remotely [9]. In 2018, the Keen Security Lab revealed several vulnerabilities in BMW cars that allow cyber-attackers to inject unified diagnostic services (UDS) packets into the CAN network, evading the central gateway [10]. Moreover, in 2020, a Toyota Lexus was the target of an attack by exploiting a Bluetooth vulnerability, causing unexpected physical motions in the vehicle [11].

Such attacks not only raise concerns about information security and privacy but also directly impact the safety of drivers, passengers, and the surrounding area. Therefore, the security of the CAN bus has become a topic of significant research interest. According to McKinsey's analysis in [12], almost all newly sold vehicles worldwide, approximately 95%, will feature connectivity by 2030, underscoring the critical need for robust CAN bus security measures. It has been proven that IDSs are an effective method for identifying cyberattacks on in-vehicle networks. An IDS monitors and identifies malicious activity on the network. In the context of in-vehicle networks, the IDS is often installed in an ECU and receives and analyzes incoming network traffic. If any abnormal messages are identified, it will notify other ECUs. In computer network systems, IDSs are used to detect and prevent intrusions. However, many conventional network security measures cannot be applied directly to in-vehicle networks. Therefore, there is a critical need for a robust IDS for in-vehicle networks.

Many studies have developed various in-vehicle IDSs using machine learning (ML) approaches. However, existing approaches overlook three crucial aspects of in-vehicle IDS requirements: robustness, limited computing resources, and deployment environment. Robustness in an IDS is the ability to detect seen attacks while staying ahead of attackers by identifying new, unseen attacks that the model has not been trained on. Ensuring the robustness of the IDS will significantly enhance the safety and security of in-vehicle networks. To design lightweight IDSs deployable in resource-constrained environments, many existing solutions rely solely on CAN ID features or use traditional ML approaches with manual feature extraction, neglecting other exploitable features and thus compromising security. Moreover, various researchers have deployed their IDSs using a traditional centralized learning approach, which involves transmitting large volumes of data to the cloud for training. This approach raises significant issues, including privacy concerns, high communication overhead, and longer response times. This paper aims to address these gaps by proposing a robust and lightweight in-vehicle IDS capable of defending against both seen and new, unseen attacks, and which can be deployed in a federated learning (FL) environment. To do so, we have laid down concrete objectives:

- Identify the limitations of existing in-vehicle IDSs and design a novel IDS to address these gaps.
- Understand CAN bus data to develop a hybrid IDS.
- Conduct data preprocessing for DL applications.
- Evaluate the hybrid IDS for both seen and unseen attacks, considering the model size.
- Compare the model's performance against previous studies.
- Propose a hierarchical federated learning (H-FL) IDS to increase the robustness of the IDS.

1.1. Contributions

This paper proposes a novel multi-stage IDS for in-vehicle network security capable of detecting both seen and unseen attacks. Furthermore, to take advantage of diverse driving scenarios and behaviors across different locations while ensuring user privacy protection, we propose a novel theoretical framework for deploying the proposed IDS in an H-FL environment. The main contributions can be summarized as follows:

- To provide a literature review on the current research on ML approaches capable of detecting seen and unseen attacks, and to identify the limitations of existing in-vehicle IDSs.
- It examines CAN bus data to extract valuable insights, aiding researchers in enhancing their understanding and improving security solutions in this field.
- It proposes a robust multi-stage IDS to detect seen and unseen malicious traffic using DL algorithms, achieving an F1-score exceeding 0.95 and a detection rate (DR) of 99.99% for previously unseen attacks.
- We have developed a novel method for detecting unseen attacks by utilizing an LSTM-autoencoder. This approach captures the sequential dependencies within a single CAN message, encompassing both the CAN ID and its payload.
- Focusing on the size, the proposed novel hybrid model is lightweight at 2.98 MB, demonstrating its feasibility for real-world deployment.
- This paper proposes a novel framework for deploying the proposed IDS in an H-FL environment.

This paper advances the field of in-vehicle IDSs by addressing key limitations of existing solutions. It leverages the power of DL algorithms and integrates two stages of detection to enhance robustness, deploying the IDS in an H-FL environment. These improvements set a new standard for further advancements toward achieving an optimally secure in-vehicle network, making it more difficult for attacks to succeed. Moreover, our analysis of CAN bus data provides valuable insights for researchers, aiding in the development of more effective security measures for in-vehicle networks. Our work paves the way for future research to explore advanced DL models, ultimately enhancing the security and reliability of CAVs. By pushing existing boundaries, we seek to contribute to a securer and more resilient automotive cybersecurity landscape. To the best of our knowledge, this is the first study to utilize a hybrid approach in in-vehicle IDSs to detect seen and unseen attacks using DL algorithms rather than traditional ML algorithms and to propose deploying the in-vehicle IDS in an H-FL.

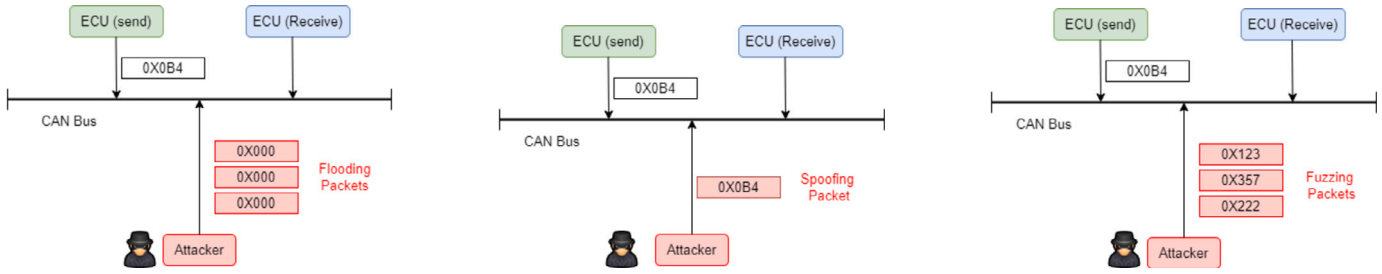
The remainder of this paper is organized as follows. Section 2 presents the background of the CAN bus protocol and attack methods. Section 3 discusses the related work. Section 4 presents the methodology of the proposed IDS. Section 5 presents and evaluates the experimental results. Section 6 provides discussions. Finally, Section 7 concludes the paper and presents potential directions for future work.

2. Background and attack methods

In this section, we provide background information on CAN bus protocol security and attack methods.

2.1. Controller area network protocol

The CAN bus is the primary communication protocol between multiple ECUs in in-vehicle networks. Robert Bosch developed the CAN bus protocol in 1985 to reduce the weight of wires, complexity, and cost [13]. Due to its high speed and efficiency, the CAN bus is widely used as the default communication standard for in-vehicle communication systems [14]. The CAN bus is a message-based broadcast protocol where ECUs transmit data in a predefined data frame as messages to all ECUs. Despite its importance, the CAN bus protocol lacks security features in



(a) DoS Attack (b) Spoofing Attack (c) Frame Fuzzification Attack

Fig. 1. CAN Bus Attacks

its design, making it vulnerable to attacks on confidentiality, integrity, and availability [3,15,16].

2.2. Attack methods

As mentioned in the 2.1 subsection, the CAN bus is vulnerable to various cyberattacks due to its broadcast nature and lack of built-in encryption or authentication [3]. To launch an attack, the attacker must first gain access to the CAN bus system, which is possible through different methods like the OBD-II port or wireless technologies [16,6]. Having successfully gained access, the attacker can inject harmful messages and launch various attacks such as DoS, spoofing, and frame fuzzification attacks. These attacks are depicted in Fig. 1.

Denial-of-Service Attack (DoS): The goal of a DoS attack is to consume the CAN bus bandwidth by transmitting a large volume of messages, which may result in unexpected system behavior. As the identifier sets the message priority, the attacker sends many messages with identifier = 0 to the CAN bus (see Fig. 1a). This message will have the highest priority [14].

Spoofing Attack: In this type of attack, an unauthorized attacker targets specific CAN IDs and tries to inject fabricated messages to control particular functions (see Fig. 1b). Because the CAN IDs are spoofed and look legitimate, it becomes challenging to distinguish between legitimate and fabricated messages, leading to system malfunction [14].

Frame Fuzzification Attack: An attacker injects random messages, which appear to be legitimate traffic, into the CAN bus network system (see Fig. 1c). Frame fuzzification attacks can compromise the ECUs and cause unexpected behavior in the vehicle, leading to problems like steering wheel shaking, unpredictable signal light on/off switching, and automatic gear shifts [17].

3. Related work

This section reviews state-of-the-art research focused on the development of in-vehicle IDSs capable of detecting both seen and unseen attacks, and their limitations. It also reviews existing federated-based in-vehicle IDSs, examines the broader impact of in-vehicle network security on urban transportation systems, and highlights the novelty of our research within this context.

3.1. In-vehicle IDSs and limitations

Research on developing in-vehicle IDSs has significantly increased over the past few years due to the critical need to enhance the security of in-vehicle networks and identify cyberattacks. Researchers have explored a wide range of approaches to developing these systems. In this paper, we focus on studies that develop in-vehicle IDSs using DL and ML approaches. An IDS can be either a signature-based or an anomaly-based system [18]. Some studies have focused on signature-based methods to detect and classify seen attacks [3,19,20], while others have employed anomaly detection methods to identify unseen or zero-day attacks [21,22]. To further improve the robustness and detection capability of the in-vehicle IDS, a few papers have developed an in-vehicle IDS that can identify both seen and unseen attacks [23,14,24,25], demonstrating significant advancements in this critical area of cybersecurity. These studies and their limitations are discussed below and summarized in Table 1.

Zhang et al. [23] have proposed a DNN-based IDS that aims to automatically extract features for the IDS from the vehicle's data packets. The authors applied gradient descent with momentum (GDM) and gradient descent with momentum and adaptive gain (GDM/AG) techniques. The study's results demonstrate the model's capability to detect replay attacks effectively. The authors accessed the sensor readings, using them as separate features. However, this method demands either having access to the CAN database (DBC) file or having knowledge about the CAN payload.

Hoang et al. [14] and Seo et al. [24] have showcased their IDSs' ability to detect both seen and unseen attacks. However, their IDSs mainly rely on the CAN ID as a singular feature; selecting only the CAN ID feature will limit the detection ability to detect attacks that involve payload manipulation [15].

Hoang et al. [14] propose a lightweight, semi-supervised, learning-based IDS to detect in-vehicle network attacks. The proposed IDS in their study integrates two DL models: autoencoders and generative adversarial networks (GAN). Their IDS was trained on unlabeled data to learn the patterns of normal and malicious data. Only a few labeled samples were used during the subsequent supervised training phase.

Seo et al. [24] have developed a GAN-based IDS (GIDS) for in-vehicle network security. The proposed IDS was trained by solely utilizing the patterns of CAN IDs from CAN data and then converting the extracted CAN IDs into a simple image. GIDS has two discriminative models to detect both seen and unseen attack data. The first discriminator is specifically trained to handle attacks. In contrast, the second discriminator and the generator are co-trained through an adversarial process. While the generator generates modified images, the second discriminator receives both modified and real CAN images, and its role is to differentiate between the modified and real images. Nevertheless, using the CAN ID as the only feature limits the detection capability of payload manipulation attacks.

The aforementioned studies employ either supervised learning-based methods or unsupervised learning-based methods. Supervised learning-based methods are frequently used to develop models distinguishing between normal and attack traffic. This technique focuses on optimizing the decision boundary to minimize classification errors on the training set and demonstrate good generalization capabilities on the testing set. Despite the high accuracy and low false alarm rate (FAR) of these models, they rely heavily on well-labeled and balanced datasets to reach their full potential, which is often difficult to achieve in practical situations. Additionally, because attackers continuously attempt to evade detection and use new, previously unseen attacks, supervised learning-based models may struggle to recognize unfamiliar attack patterns not present in the training data [27]. This limitation can lead to significant security consequences. Unsupervised learning-based methods, meanwhile, build models exclusively using normal data, relying on profiling

Table 1
Related Works of ML-based IDSs for in-vehicle Network

Ref	Year	Category	Algorithm	Dataset	Seen Attacks	Unseen Attacks	M-C	ID-based Detection	Payload-based Detection	DL	FL
[23]	2019	Supervised	DNN	Simulation	✓	✓		✓	✓	✓	
[14]	2022	Semi-supervised	AE, GAN	Car-Hacking [24]	✓	✓		✓		✓	
[24]	2018	Unsupervised	GAN	Car-Hacking [24]	✓	✓		✓		✓	
[25]	2022	Hybrid	DT, RF, ET, XGBoost, CL-k-means	Car-Hacking [24], CICIDS2017 [26]	✓	✓	✓	✓	✓		
Our work	**2023**	**Hybrid**	**ANN, LSTM-AE**	**Car-Hacking [24]**	✓	✓	✓	✓	✓	✓	✓

DL: Deap Learning, **FL:** Federated Learning, **M-C:** Multi-classification.

Table 2
Constant and Changing Data in CAN Message

CAN ID	D0	D1	D2	D3	D4	D5	D6	D7
399	254	59	00	00	00	60	00	00
399	254	60	00	00	00	61	00	00
.	.	.	00	00	00	.	00	00
.	.	.	00	00	00	.	00	00
.	.	.	00	00	00	.	00	00
399	254	94	00	00	00	74	00	00
399	254	95	00	00	00	75	00	00

normal traffic behaviors to detect anomalous traffic that could indicate a potential attack. As a result, unsupervised learning-based models are well suited to detecting previously unseen attacks [28].

To leverage the strengths of both approaches, Yang et al. [25] have developed a multi-tiered hybrid IDS, MTH -IDS, to protect intra-vehicle and external networks from cyberattacks. MTH-IDS combines supervised and unsupervised models. Despite achieving good results, the proposed IDS has certain limitations that we aim to address.

One significant limitation to note is that the authors used only four features—CAN ID, D5, D3, and D1—to train the model after feature extraction. While this strategy may result in a more efficient model, it does create the potential for an attacker to manipulate other features that were not considered during the model's training process [29]. This is a crucial limitation in CAN bus data for three reasons. First, selecting a subset of features from the CAN bus payload as important features and discarding non-important features could pose a risk, as attackers might exploit the neglected features, effectively bypassing the model [30,31]. Second, the evolving landscape of attack scenarios means that features chosen to detect one category of attacks may become outdated or insufficient for addressing new, unseen attacks [29]. Third, upon closer examination of the CAN bus data, we noticed that each CAN ID exhibits distinct patterns of data field values. For example, CAN ID 399 consistently has zero values in data fields D2, D3, D4, D6, and D7, which remain unchanged. By contrast, in CAN ID 320, the data fields that are always 0 are D1, D2, and D3. Table 2 displays the constant zero values in blue, indicating data that never changes throughout the dataset. Data highlighted in yellow represents changing values, while data highlighted in green denotes constant non-zero values. Consequently, features that are significant (possessing varying values) for one CAN ID may not hold relevance for another, making the selection of a consistent subset of important features for all CAN IDs impractical due to the inherent nature of CAN bus data. This observation can help researchers understand the CAN bus data and make their IDS models more focused on the most significant features, potentially improving their generalization capability.

Another limitation pertains to the deployment approach. Yang et al. suggest conducting the training process on a server machine and the testing process on the CAN bus. However, this deployment approach requires training the data on a remote server, which involves sending sensitive data to a server for training, raising privacy concerns.

In the unsupervised model, the F1-score of the proposed MTH-IDS was only 0.82. To improve this result, the authors add another tier of two biased classifiers. Training these biased classifiers on specific errors—false positives (FPs) and false negatives (FNs)—may lead to poorer performance when we test the model on new, unseen data. Additionally, adding this tier transforms the model from being purely unsupervised. Resulting in a dependency on labeled datasets, which are often difficult to implement in practical situations.

A key distinction between MTH-IDS and our approach is our use of DL algorithms over conventional ML models. Multiple authors have found that DL-based IDSs outperform traditional ML IDSs in automotive applications [32]. This superiority is due to several factors: DL methods are more adaptive, continually being refined with incoming data, which is particularly suitable for the nature of CAN bus data [23]. Additionally, traditional ML often requires manual feature engineering, such as applying correlation-based feature selection, which can be time-consuming [33]. In contrast, DL automatically deduces features, allowing algorithms to directly discern optimal features from raw data [34]. Furthermore, DL IDSs are especially capable of detecting novel attacks and can scale more effectively to highly complex in-vehicle network data while maintaining efficacy [34]. Therefore, there is a need to develop a robust, lightweight in-vehicle IDS that addresses previous limitations.

3.2. Federated-based in-vehicle IDSs

FL is a privacy-preserving decentralized learning technique that trains models locally without transferring row data to a centralized server. Instead, it transfers model parameters to a centralized server, which aggregates the clients' models to build a shared global model [35]. The integration of FL into IDSs addresses the growing need for enhanced security and privacy in our interconnected society. Although some previous works [36–41] have deployed their in-vehicle IDSs in an FL environment, all the proposed FL works used a standard (non-hierarchical) architecture, relying solely on one central server (aggregator). However, this can introduce performance challenges, particularly due to delays in sharing the model between the central aggregator and numerous devices. Furthermore, this central server poses a risk as a single point of failure [42] and is restricted to limited driving behaviors under its coverage [43].

3.3. Impact on urban transportation systems

The impact of securing in-vehicle networks in CAVs extends to the broader field of urban transportation planning, where vehicles are integral to the overall system. Urban transportation research aims to mitigate traffic congestion [44], improve safety, increase the adoption of

renewable energy [45], and advance sustainable mobility [46]. A recent survey [47] examining the impacts of CAVs on urban transportation and the environment found that CAVs would decrease energy consumption and protect the environment by reducing emissions. Furthermore, these vehicles could significantly reduce traffic crashes involving human error while increasing the convenience and productivity of passengers. However, there are widespread concerns about personal safety, security, and privacy due to the potential for cyberattacks. Therefore, enhancing the security of CAVs would significantly contribute to the improvement of urban transportation systems.

3.4. Research novelty

While previous studies have achieved notable results in specific areas, they also have several limitations. Compared to existing studies related to in-vehicle IDS, our proposed IDS offers the following advantages: 1) It uses DL rather than traditional ML, which has proven to be more efficient in automotive applications and is capable of detecting novel attacks more effectively [32,34]. 2) It employs a hybrid model (signature-based and anomaly detection) instead of relying on a single model, which has been shown to improve detection results [48]. 3) It successfully detects both seen and new, unseen attacks. 4) Despite using DL algorithms, it is a lightweight IDS. 5) It utilizes both CAN ID and payload as features, which enables detection of CAN ID changes and payload manipulation attacks [15]. 6) It continuously learns by labeling and updating the signature-based classifier when a new, unseen attack is detected. 7) It leverages diverse driving behaviors by being deployed in an H-FL environment.

The novelty of this paper lies in three key aspects: the design of the IDS, the algorithms used, and the deployment approach. First, our proposed in-vehicle IDS employs a cascaded multi-stage approach to detect both seen and unseen attacks. Unlike other multi-stage IDS designs in the literature, our IDS uses the first stage to classify traffic into seen attacks and normal data. The second stage then re-examines the data classified as normal in the first stage to detect unseen attacks that bypassed the initial model, providing an additional layer of protection. Furthermore, the IDS continuously learns by labeling and updating the classifier in the first stage when a new, unseen attack is detected. Secondly, we use a hybrid approach (ANN and LSTM-autoencoder) that leverages DL algorithms. By doing so, our proposed IDS ensures a multi-layered defense mechanism against potential threats and improves detection performance compared to single-point IDSs [49]. Thirdly, to further enhance the robustness of the in-vehicle IDS, our approach addresses the significant drawbacks of previous methods by proposing its deployment in an H-FL environment. This approach aims to build a more robust global model that takes advantage of diverse driving scenarios and behaviors in different locations while ensuring user privacy protection.

In summary, the review of existing literature reveals several research gaps and shortcomings, highlighting the broader impact of in-vehicle network security on urban transportation systems and emphasizing the need to improve the security of these networks. These gaps provide a critical foundation for the design and novelty of our proposed IDS.

4. Methodology

In this section, we present the methodologies used to develop our proposed in-vehicle IDS. First, we explain the proposed in-vehicle IDS, including data preprocessing; the first stage, a supervised model (ANNs); and the second stage, an unsupervised model (LSTM-autoencoder). Next, we provide details on training and testing the model, including ANN hyperparameter tuning and LSTM-autoencoder hyperparameter tuning. Finally, we present the proposed H-FL framework.

4.1. Proposed in-vehicle IDS

In Section 3, we highlighted the limitations of existing in-vehicle IDSs. To address these limitations, we introduce a multi-stage IDS designed to protect in-vehicle networks from seen and unseen attacks by using a hybrid approach (supervised and unsupervised algorithms). We adopt a hybrid approach to mitigate the risks inherent in relying on a single model. Integrating multiple ML models can significantly enhance performance, improve data security, and reduce the FN rate compared to using a single model [37,48]. These benefits are particularly valuable for in-vehicle networks, where errors can be costly. Additionally, our proposed hybrid approach increases protection against attackers, who must now evade two models instead of just one.

Our proposed IDS integrates both supervised and unsupervised models. As illustrated in Fig. 2, the CAN bus data first enters the data preprocessing stage, followed by the supervised classifier in the initial stage. Subsequently, the normal data is processed by the second model to identify any previously unseen attacks. The supervised model is primarily responsible for detecting and categorizing previously seen attacks based on historical data. By placing the supervised classifier model first, the IDS can quickly filter out any attacks based on its training and accelerate the detection process. The subsequent unsupervised model serves as a secondary layer of protection against unseen attacks that bypass the first model. When the supervised model makes a mistake and classifies the malicious traffic as normal, the unsupervised model can detect this and flag it as an anomaly. The unsupervised model, which works as an anomaly detection model, is trained solely on normal data, and any samples that deviate significantly from the learned patterns are identified as an anomaly or an unseen attack. Once the unsupervised model detects malicious traffic, it is flagged for further investigation. Any anomalies detected by the unsupervised model that are later confirmed as threats will have a new attack label generated. This new label will be used to further train and refine the supervised model, enabling it to recognize such attacks in the future and ensuring the system improves over time. As the vehicular environment evolves, new attack vectors may emerge. The unsupervised model ensures that the system remains adaptive and resilient in the face of changing attacks, even if the supervised model has not been trained on them. This comprehensive multi-stage IDS ensures coverage for both seen and unseen attacks. For broader applicability, the proposed IDS can learn the legitimate CAN IDs and normal behavior for each vehicle at design time and monitor the network to detect any attacks during operational runtime.

Our proposed IDS provides a comprehensive and robust solution to secure in-vehicle networks. Furthermore, our IDS addresses significant challenges associated with previous approaches by deploying it in a FL environment. Utilizing FL allows the IDS to benefit from various driving scenarios, enhances its resilience against new and unseen attacks, and enables continuous learning without compromising the privacy and security of the training data. With the rapid development of in-vehicle technologies and communication protocols, having a resilient IDS ensures that we are prepared for both current and future threats.

4.1.1. Data preprocessing

In data preprocessing, data was converted into a format suitable for use by deep neural networks. This was achieved by applying different operations. Fig. 3 shows the data processing applied to each feature in the dataset. The dataset contained four files, each corresponding to a specific attack (DoS, frame fuzzification, gear, and RPM), with both attack and normal instances. Table 3 shows the number of attacks and normal instances in each file. We processed each file individually and then concatenated them into a single dataset. We shifted the flag field to the last column and filled non-available data bytes with NAN values. Given the extensive data points we had, we removed any row with these missing values in the data fields. The CAN ID and data fields were in hexadecimal values. We converted the CAN ID and data values from hexadecimal to decimal values as per the ML requirement. Most ML algorithms require the conversion of categorical data into numerical data. Therefore, after concatenating the datasets into one dataset, a label encoder was employed to convert categorical features in the flag feature

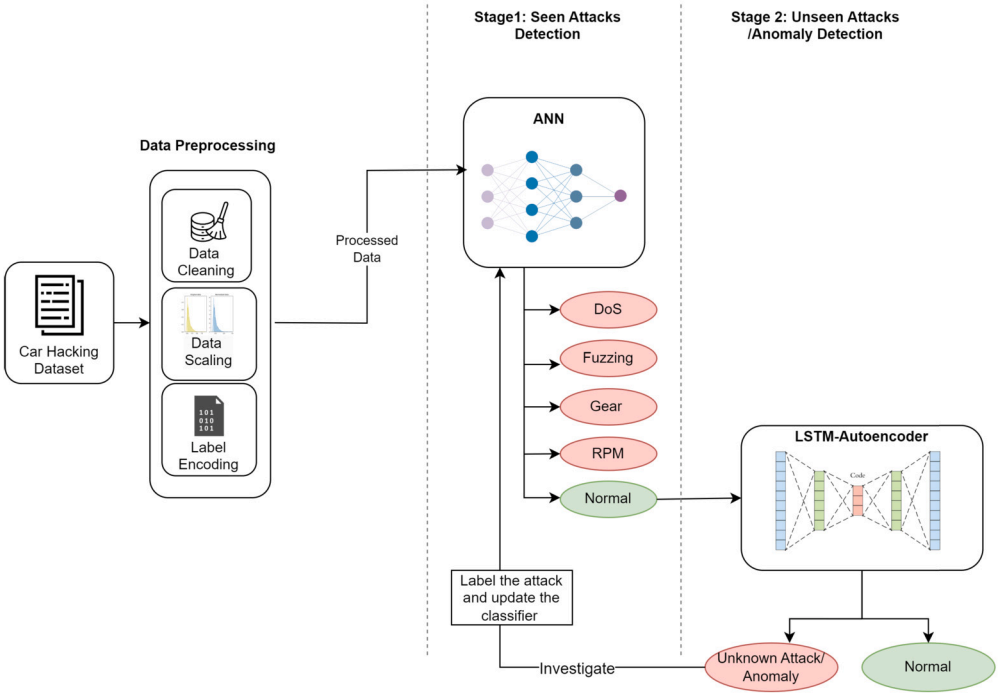


Fig. 2. Workflow Model Depiction of Proposed Multi-stage IDS for In-vehicle Network

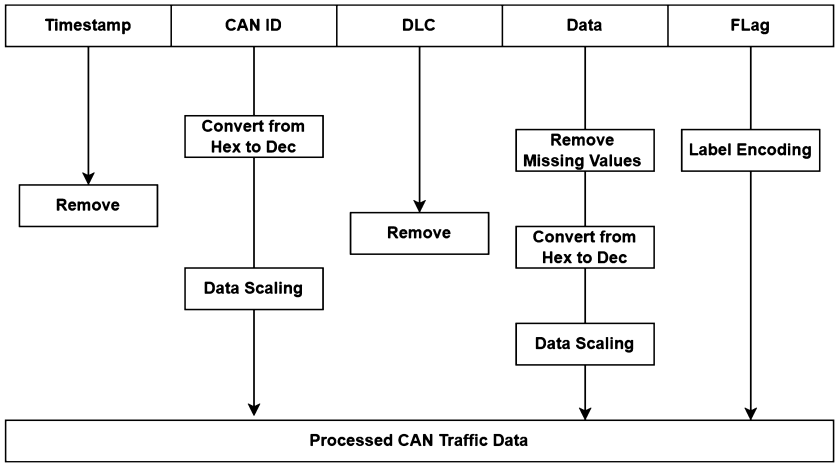


Fig. 3. Data Preprocessing

(Normal, DoS, frame fuzzification, gear, and RPM) into numerical representations to facilitate their use as inputs in ML algorithms.

As shown in Table 3, the dataset comprised millions of data points and was highly imbalanced, with the number of attacks accounting for 14.06% and normal data making up 85.94%. This could result in high processing times and produce biased models. To mitigate issues related to the dataset's size and model complexity, data sampling is a common approach used to generate a representative sample from the original data [50]. After reviewing and experimenting with various data sampling methods, we adopted the same sampling approach used in [25]. In their work, they employed the K-means clustering algorithm for data sampling and then addressed class imbalance issues using the synthetic minority oversampling technique (SMOTE). Fig. 4 depicts the difference between the number of samples in the original data and the number after applying sampling and SMOTE to balance the data. Although we trained the models on sampled data, we tested them on the remaining dataset to ensure that the models were well generalized and capable of recognizing the entire dataset. We used this method only for unsupervised anomaly detection because the supervised classifier handled the entire dataset efficiently and quickly.

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 (14.06%)	14,237,958 (85.94%)

We considered nine columns (CAN ID and eight data values) as the features in our experiment. This choice was made to effectively detect any manipulation of either the CAN ID or the payload. We opted to include the CAN ID feature because the utilized dataset [24] was created by injecting attacks into arbitrary CAN IDs, which made incorporating this feature crucial for capturing the distinct characteristics of cyberattacks. To avoid biased models and potentially inaccurate results due to

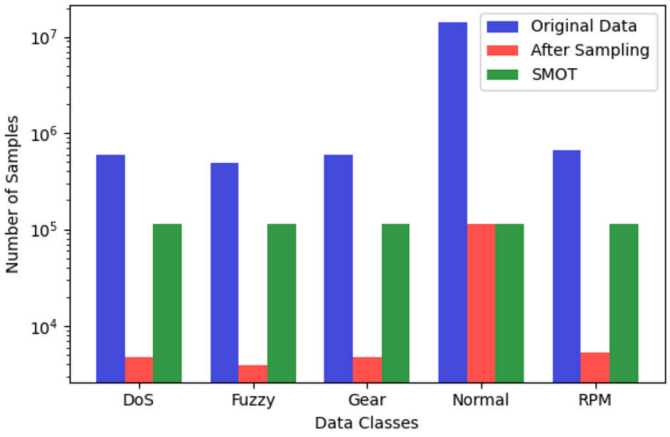


Fig. 4. Comparison between the number of data classes before and after sampling and balancing data

Table 4
Data Features and Types Before and After Data Preprocessing

Feature	Before	After
CAN ID	hexadecimal	decimal integer
D0	hexadecimal	decimal integer
D1	hexadecimal	decimal integer
D2	hexadecimal	decimal integer
D3	hexadecimal	decimal integer
D4	hexadecimal	decimal integer
D5	hexadecimal	decimal integer
D6	hexadecimal	decimal integer
D7	hexadecimal	decimal integer
Flag	string (T, R)	decimal integer (0,1,2,3,4)

the clear correlation between timestamps and cyberattack simulation intervals, the timestamp feature was intentionally excluded. In addition, we excluded the data length code (DLC) feature since it could be correlated with the data fields, given that DLC represents the payload's length. To decrease the feature dimensions, we decided not to consider DLC. We then applied the StandardScaler algorithm to scale the data. In comparison to other algorithms, StandardScaler was able to find a better balance between handling outliers and preserving the range of values. It standardizes data by subtracting the mean value, resulting in a zero mean, and then dividing by the variance, thus giving the distribution unit variance, as shown in Equation (1):

$$X_{\text{scaled}} = (X - X_{mean})/X_{std} \tag{1}$$

Where, X_{scaled} represents the scaled value, X is the original value, and X_{mean} and X_{std} denote the mean and standard deviation, respectively.

This step is important because network traffic data can possess differing ranges, and normalized (scaled) datasets tend to enhance the performance of ML models [51]. Failure to normalize the dataset, especially when its features have different scales, may cause the ML model to concentrate primarily on the features with larger scales [25]. Table 4 shows the data features and types before and after data preprocessing.

4.1.2. First stage: supervised model

We reviewed various DL algorithms for attack detection and multi-classification and concluded that artificial neural networks (ANNs) were the optimal choice for our data for the following reasons:

1. ANNs introduce non-linearity into the model, which makes them capable of modeling complex patterns and relationships that linear classifiers might miss.
2. Unlike traditional methods, ANNs can automatically learn the best features directly from the raw data without requiring explicit feature engineering. This can lead to better performance, especially in CAN bus data, where each CAN ID has distinctively informative features.
3. ANNs are adaptive systems, making them suitable in situations where data is continuously evolving.
4. ANNs can produce highly competitive results.

However, while ANNs can be an excellent choice, they can be computationally intensive. To address this, we simplified the ANN architecture by reducing the number of layers and neurons while still achieving the highest DR.

4.1.3. Artificial neural networks

ANNs are ML algorithms inspired by the behavior of biological neurons in the brain and the central nervous system [52,53]. ANNs' inputs pass through one or more hidden layers, assign weights, and produce an output. Fig. 5 depicts the ANN architecture. ANNs can adjust their internal parameters, known as weights and biases, for both the hidden and output layers. This adaptive feature means that ANNs can understand the deep and non-linear interrelations between dependent and independent variables without any prior knowledge [54]. In contrast to traditional classification algorithms that often demand knowledge of the system's probability model, ANNs operate as a "black box" that can adapt to the underlying system model [55]. Their adaptability, particularly in high-dimensional datasets, addresses challenges associated with conventional algorithms such as k-nearest neighbor and decision trees [56]. Across various application domains, ANNs have been employed for classification tasks and have also demonstrated their efficacy in several computer security areas, including detecting network attacks [57,58].

4.1.4. Second stage: unsupervised model

The second stage of the proposed multi-stage IDS is an unsupervised model that acts as an anomaly detector, identifying anomalies or unseen attacks that bypass the first stage. Unsupervised learning-based models are particularly well suited to detecting previously unseen attacks [28]. We selected the LSTM-autoencoder model as the anomaly detector model for in-vehicle IDS for several reasons. Firstly, as an unsupervised neural network, the autoencoder does not require labeled data, saving significant time and effort. Secondly, research shows that autoencoders are effective at detecting anomalies and unseen attacks, making them well suited for in-vehicle IDS [14,22,59]. Lastly, LSTM layers capture temporal dependencies within sequential data, a feature that conventional autoencoders lack. These features make the LSTM-autoencoder model a robust and efficient solution for detecting anomalies and enhancing the security of in-vehicle networks.

4.1.5. LSTM-autoencoder

The Long Short-Term Memory (LSTM) autoencoder consists of both LSTM and autoencoder components. LSTM, a type of recurrent neural network (RNN), is designed to handle sequential data and can learn complex dynamics within the temporal order of input sequences by using internal memory to store information over long sequences. This capability is particularly useful for CAN bus data, which is sequential [15]. In contrast, autoencoders are neural network architectures designed to learn efficient representations of input data by attempting to reconstruct the original data as accurately as possible. By combining LSTM with an autoencoder, we aimed to capture the sequence order in each CAN bus message, which classic autoencoders might overlook. The chosen LSTM-autoencoder consists of two interconnected LSTMs: the first encodes sequences of features into a fixed-size vector, and the second decodes this vector back into a sequence. In the context of detecting anomalies in the CAN bus, the LSTM-autoencoder was trained only on normal data. This allowed it to accurately learn to reconstruct the benign patterns it was trained on. When test data was presented, input data was first encoded by the LSTM into a fixed-size vector. Another LSTM

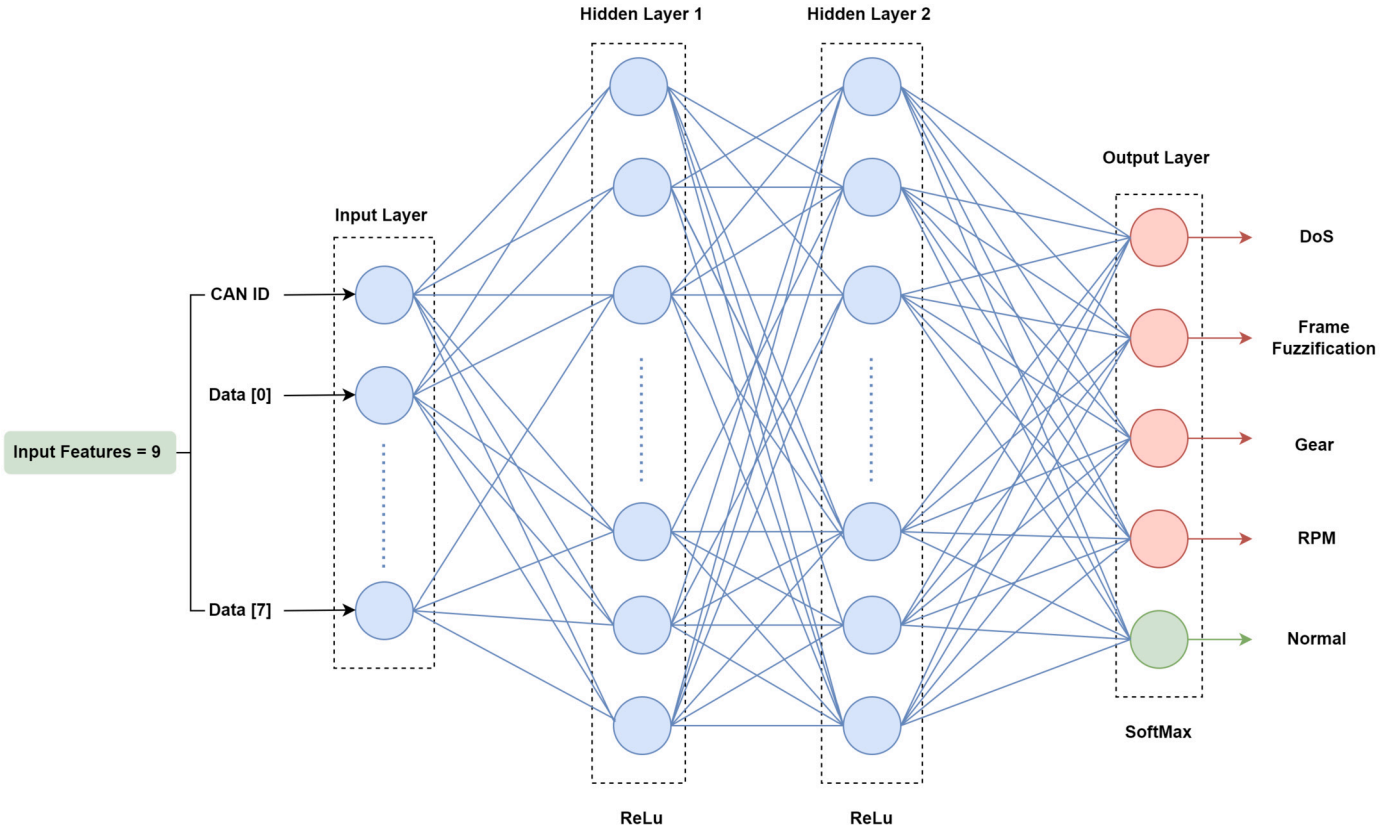


Fig. 5. ANN Architecture

then decoded this vector to reconstruct the input data. Any deviations in reconstruction can be potential indicators of anomalies.

The motivation for using the LSTM-autoencoder in this context arose from the sequential nature of the CAN ID and its associated data fields. Incorporating both the CAN ID and its data fields (payload) is essential because normal data fields for one CAN ID might be considered unusual for another CAN ID. The LSTM-autoencoder was trained on normal sequences, with any deviation from these established sequences subsequently flagged as anomalous. However, it is worth noting that several studies have used sequence-based models to explore the sequential dependencies between CAN IDs only or multiple CAN bus messages to detect anomalies. However, to the best of our knowledge, there has been no research examining the order dependence within a single CAN message, including both the CAN ID and its payload. This is important because CAN IDs can follow periodic or event-driven patterns, and sequences of CAN IDs can change when event-triggered messages occur on the CAN bus [15]. For example, the sequence of CAN IDs or CAN messages can change when a sudden event happens, such as someone opening the door. We trained an LSTM-autoencoder model to reconstruct benign sequences with minimal errors, expecting the model's inputs and outputs to look alike. However, when a malicious sequence was fed into the model, the model was expected to fail at reconstructing the sequence. Therefore, the input and output vectors were expected to differ significantly. Fig. 6 depicts the architecture of the LSTM-autoencoder.

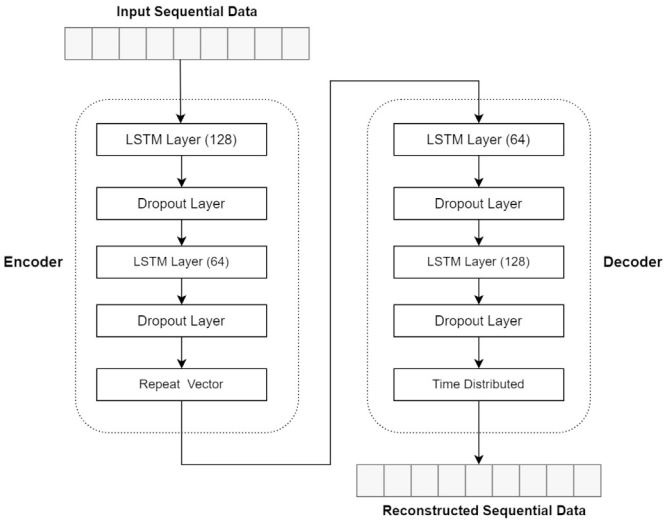


Fig. 6. LSTM-Autoencoder Architecture

4.2. Training and testing the model

To prevent overfitting in ML models, we applied the data partitioning method outlined in [60] by dedicating 70% to training and the subsequent 30% to testing the model's performance. In the ANN model, we trained the model using labeled data to learn the relationships between the features and the targets. Then, we tested the model on test data to assess the quality of the model. The ANN works as a multi-class classifier, and the output will be normal or attack type, including DoS, frame fuzzification, RPM, and gear spoofing. Further details regarding these attacks are discussed in Section 5.2. The data classified as normal in the first stage of the IDS (ANN) served as input for the second stage (the LSTM-autoencoder). Our experiment involved adjusting multiple hyper-parameters, such as batch size, number of units, learning rates, optimizers, activation, and loss functions. Through systematic experimentation, we identified the most suitable hyper-parameter values that could deliver the best possible performance and efficiency.

Table 5
ANN Parameters for Multi-classification

Parameter	Value
Epoch	10
Number of Hidden Layers Neurons	16
Number of Output Layer neurons	5
Number of Hidden Layers	2
Input Layer Activation Function	ReLU
Hidden Layer Activation Function	ReLU
Output Layer Activation Function	softmax
Optimizer	Adam
Batch Size	256
Shuffle	True
Loss Function	categorical_crossentropy

4.3. ANN hyperparameter tuning

The model was a feedforward ANN that comprised four layers: an input layer, two hidden layers, and an output layer. It was designed for multi-class classification, targeting five distinct classes (DoS, frame fuzzification, RPM, gear spoofing, or normal). We employed a grid search approach to systematically explore and identify the optimal hyperparameters for the ANN classifier. The first layer was the input layer, with an input shape of nine. The next two layers were hidden layers, which are dense layers, each utilizing 16 neurons and the ReLU activation function. The ReLU function was preferred in hidden layers because it introduced non-linearity to the model. The output layer comprised five neurons, corresponding to the number of output labels. For this layer, we employed a softmax activation function, which is suitable for multi-class classification problems. Considering the one-hot encoded labels, we chose the categorical cross-entropy loss function for compiling our model. We optimized the model with the Adam optimizer, maintaining the default values for other parameters. To enhance training efficiency and mitigate overfitting, we integrated an EarlyStopping callback. Table 5 summarizes the hyper-parameters and their respective values used in tuning the ANN model.

4.4. LSTM-autoencoder hyperparameter tuning

In the LSTM-autoencoder model provided, hyperparameters were carefully selected for optimal training. However, before feeding the data into the LSTM-autoencoder, we reshaped the data from a 2D format of (Samples, Features) to a 3D format of (Samples, Time Steps, Features) by using the reshape function. This was because the LSTM requires a 3D input shape. The model operated on sequences of length time_steps, which was set at 1, meaning each data point was treated as a sequence of its own. Each of these sequences had nine features, which included the CAN ID and eight other data fields from the CAN message. This LSTM-autoencoder structure consists of an input layer, two encoder layers, a RepeatVector layer, two decoder layers, and an output layer. The first LSTM layer, which serves as both the input and the first encoder layer, has 128 neurons and is designed to return sequences, enabling the stacking of subsequent LSTM layers. The second LSTM encoder layer, with 64 neurons, does not return sequences. Following the RepeatVector layer, which replicates its input features, the LSTM decoder consists of two LSTM layers with 64 and 128 neurons, respectively. The activation function employed in the input and hidden layers was ReLU, chosen for its rapid and efficient training on large datasets compared to the sigmoid function. Conversely, the last decoder layer employs sigmoid activation function. To combat overfitting, four dropout layers with a dropout rate of 20% were instituted after each LSTM layer, randomly deactivating one-fifth of the neurons during every training iteration. We used a batch size of 64 and 100 epochs. Adam optimizer was used along with mean squared error (MSE) loss function, which is known for its application in anomaly detection problems. To enhance training efficiency and to avoid overfitting, we integrated an early-stopping callback and dropout layers.

Table 6
LSTM-Autoencoder Parameters for Binary Classification

Parameter	Value
Epoch	100
Input Layer Activation Function	ReLu
Hidden Layers Activation Function	ReLu
Optimizer	Adam
Batch Size	64
Dropout Rate	0.2
Shuffle	True
Loss Function	MSE

Table 6 shows the parameters and their respective values in hyper-tuning the LSTM-autoencoder. We trained the LSTM-autoencoder using only the normal data labeled in the dataset as 0, but it was tested using both normal and attack data.

In our anomaly detection method using the LSTM-autoencoder, the model's capability to reconstruct input data was crucial for detecting anomalies. The reconstruction errors were calculated using the MSE between the reconstructed output and the original data. To delineate a boundary for what qualifies as an anomaly, a threshold was established. Determining the optimal threshold for anomaly detection can be intricate. Through various experiments, we found the most effective threshold is the sum of the average reconstruction error and one standard deviation of these errors from the training set. In simpler terms, the threshold creates a margin above the average error, and any data point with an error exceeding this margin will be considered anomalous. Then, the model can predict anomalies by comparing each test's reconstruction error to the pre-defined threshold. The threshold is shown in Equation (2):

$$\text{Threshold} = \mu(\text{train\_errors}) + \sigma(\text{train\_errors}) \tag{2}$$

In this context, μ represents the mean, while σ indicates the standard deviation.

4.5. Proposed hierarchical federated learning-based framework

In this section, we introduce the concept of H-FL, which served as the core technique for enabling our proposed IDS to leverage diverse driving scenarios and behaviors across different locations while ensuring user privacy protection.

Previous researchers have deployed in-vehicle DSs using a traditional centralized learning approach, which involves transmitting large volumes of data to the cloud for training. However, this approach raises privacy concerns and involves high communication overhead and longer response times [36]. Therefore, recent research has shown a growing interest in adopting the FL approach to address these issues.

FL is a unique implementation of a distributed ML approach that involves training a model on edge devices (clients) without transferring the raw data to a central location [61]. Thus, FL is an ideal fit for in-vehicle IDSs for several compelling reasons. First, the FL approach preserves data privacy since the learned parameters from local models are sent periodically to the cloud server instead of transmitting the whole row of data. Second, FL allows multiple participants to develop a robust and efficient global model without compromising user data privacy. This feature makes it a better option than non-FL approaches [62], offering real-time model updates and allowing access to data without contacting the centralized server. Third, FL reduces latency by avoiding transmitting the raw data to a central server. This is crucial, as sending data to a central server can be costly and may impair the effectiveness of IDS deployments [63]. Fourth, FL's distributed nature and low complexity make it highly suitable for resource-constrained hardware deployments [37]. Based on the guidelines provided by the International Telecommunication Union for IDS in vehicular networks in 2020 [64], it is essential that an in-vehicle IDS offers the capability to regularly up-

date its rule sets. Thus, FL makes the IDS adaptive to new, unseen attacks by updating local models with new models trained on new, unseen attacks identified in other clients. This adaptiveness allows the local model to stay up-to-date and more robust against new, unseen attacks. Additionally, since FL involves training models locally on devices and only sharing model updates, it ensures that sensitive data remains private and secure throughout the process.

However, the FL approach relies solely on one central aggregator, which can introduce performance challenges, particularly due to delays in sharing the model between the central aggregator and numerous devices. Moreover, this aggregator poses a risk as a single point of failure. To address these issues, we decided to deploy the proposed IDS in H-FL, which incorporates a central aggregator with multiple local aggregators instead of having one central aggregator. This approach aimed to overcome the limitations of relying solely on a central aggregator [42].

The architecture of H-FL consists of three layers: the central server, edge servers, and end devices. The end device layer comprises multiple end devices (vehicles), where each end device has a large amount of local sensor data and a locally installed model. In the edge server layer, multiple local aggregators act as intermediaries for communication between end devices and the central server. Their primary roles include transferring model parameters and aggregating local model parameters. The central server layer, located in the cloud, is accountable for the distribution and continuous updates of the final global model. Fig. 7 depicts a high-level overview of the H-FL process.

At the beginning of each round, the central server initializes a global model. Then, the central server sends the global model to each local aggregator. Each local aggregator receives the global model, selects a subset of end devices using a selection approach to participate in the training process, and then dispatches the global model to the selected end devices. As CAN bus data can vary significantly between vehicle makes and models, we propose that the local aggregator selects a subset of available end devices (vehicles) based on the similarity in CAN bus data, such as make or model, in specific geographical areas. Grouping vehicles by similarity in CAN bus data ensures that the global model is trained on similar and relevant data, yielding more accurate and reliable results. Once the global model is deployed on each end device, it leverages its local data for training in small batches. After local training, end devices send their trained local model parameters to the corresponding local aggregator. Each aggregator aggregates all the local model parameters/weights using an aggregation algorithm to obtain the edge aggregation model. After all local aggregators have completed the edge model aggregation, they send all the aggregated model parameters to the central server to build a new global model. These aggregated model parameters contribute to refining and improving the global model, which is then sent back to the end devices for continuous training. The central server adjusts the global model parameters based on the aggregated model parameters, preparing for the subsequent round of global training.

The training process in H-FL can be divided into three parts: local training, edge aggregation, and global aggregation. For example, when a local model identifies an anomaly or previously unseen malicious network traffic, it isolates the data for further examination. If this anomaly is determined to be a new, previously unseen attack, a new label is created in the classifier for seen attacks (the first stage in the IDS). In response, the server updates the global model and transmits the updated version to the edge aggregators and then to the selected end device and relevant end devices. After each communication round, the end device returns the model's parameters to the corresponding edge aggregator and then to the central server, allowing our proposed IDS to adapt. To keep the seen attack detection model effective, seen attack data needs to be updated regularly. It continuously monitors for unseen attacks and updates the supervised classification model upon detection, ensuring its adaptability to new, unseen attacks. This process is repeated until the desired level of performance is obtained.

Table 7
Data Features, Descriptions, and Types

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

5. Results and evaluations

5.1. Experiment setup

The implementation was performed in Google Colab Pro, a web-based editor from Google Research that allows users to write and run arbitrary Python code from the browser. The experimental setup consisted of a 64-bit Ubuntu 20.04.5 LTS operating system, 11th Gen Intel(R) Core(TM) i7-11700 @ 2.50GHz, 31.1GiB RAM, NVIDIA Corporation, and Python 3.9.16 version.

5.2. Dataset description

To assess the performance of our proposed model, we utilized a benchmark dataset published by Song et al. [24]. This dataset is extensively used in automotive security research and comprises four types of attacks: DoS, frame fuzzification, engine RPM spoofing, and drive gear spoofing. We selected this dataset because it is based on real-world traffic data rather than simulated data, and it allows us to compare our proposed IDS approach with similar work that uses the same dataset [25]. For every CAN message, the dataset provides valuable information, including timestamp, CAN ID, DLC, data field, and flag. The timestamp indicates the precise time when the message was recorded from the startup. The CAN ID plays a crucial role in determining the priority of multiple messages, with lower values being given precedence over higher ones. Moreover, the DLC specifies the data field's length in bytes, up to 8 bytes. The flag indicates whether the message is normal or an attack. In Table 7, we present the dataset's features, descriptions, and data types.

To select the most suitable algorithms, we examined the CAN bus data from Car Hacking Dataset [24], ensuring a comprehensive understanding of both normal and abnormal behaviors. Attacks include:

DoS Attack: Inject many ZERO values into every CAN bus ID and payload at 0.3 millisecond intervals. This dominates the BUS, causing legitimate messages to be delayed or blocked.

Frame Fuzzification Attack: Messages are randomly injected into the CAN bus. There are two types of frame fuzzification attacks present in this dataset:

1. Injecting random IDs (not seen before).
2. Injecting IDs that appear legitimate but have a different payload.

During a frame fuzzification attack, an adversary might expect some valid CAN messages to inadvertently cause a malfunction in the target vehicle. It is presumed that the adversary has no prior knowledge of the in-vehicle communication of the target vehicle. Thus, the adversary injects messages with random CAN IDs and payloads. This implies that both the CAN IDs observed in normal traffic and those not seen before can be included in a frame fuzzification attack.

RPM Spoofing Attack: This spoofing attack specifically targets RPM CAN ID: 790, aiming to inject fabricated messages to control various functions. The legitimate data payload is different from the fabricated messages.

Gear Spoofing Attack: This spoofing technique targets the Gear CAN ID: 1087, attempting to inject fabricated messages to control functions. While the fabricated messages resemble normal ones, they are not identical. For illustration purposes, Table 8 presents examples of both

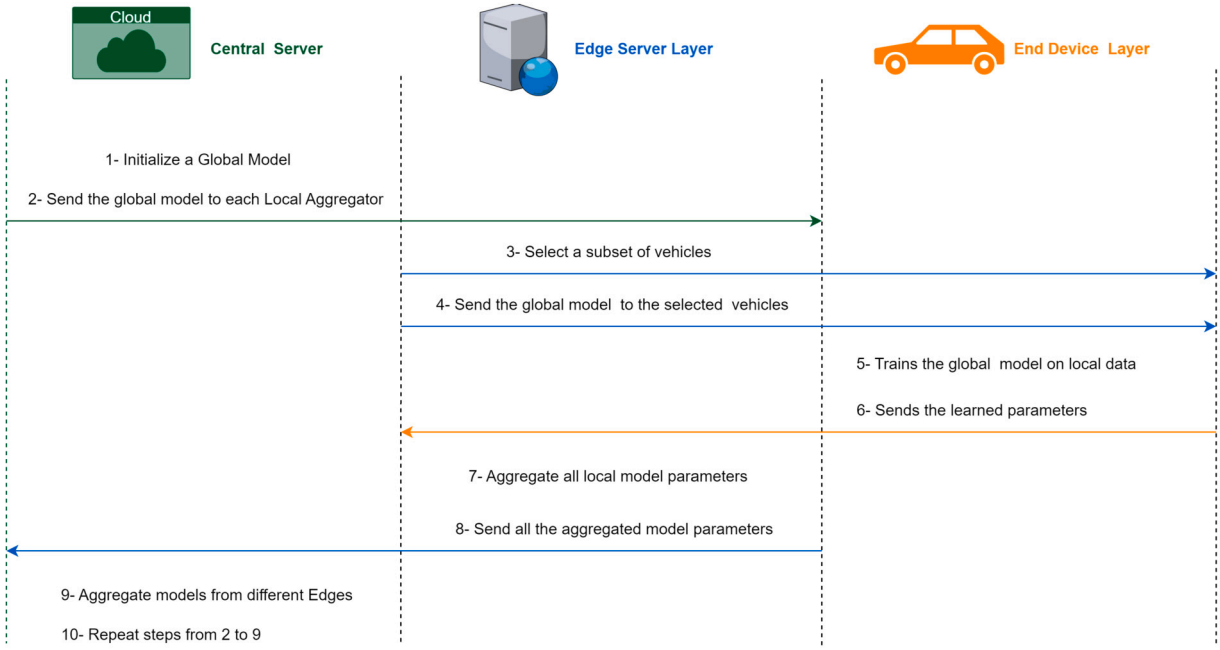


Fig. 7. A High-level Overview of H-FL Process

Table 8
Normal and Attack data in Car Hacking Dataset

CAN ID	D0	D1	D2	D3	D4	D5	D6	D7	Class
880	0	64	96	255	120	0	8	0	Normal
0	0	0	0	0	0	0	0	0	DoS
55	0	1	1	0	11	22	0	1	Frame fuzzification
880	0	0	0	255	0	0	10	0	Frame fuzzification
790	0	30	40	0	0	13	0	9	Normal
790	0	8	8	0	0	0	0	11	RPM
1087	1	2	1	0	0	1	240	7	Normal
1087	0	22	1	180	0	130	0	33	Gear

normal data and various kinds of attack data. Data represented in black indicates normal data, while data in red signifies attack data.

5.3. Evaluation metrics and performance evaluation

To assess the robustness of the proposed IDS, we considered various performance metrics such as accuracy (Acc), F1-score (F1), precision (Pre), recall (Rec)—or, as it is called, DR—and FAR. Metrics were determined based on true positive (TP), true negative (TN), FP, and FN values. We used the following equations to calculate the metrics used:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$F1 = 2 \times \frac{Pre \times Rec}{Pre + Rec} \tag{4}$$

$$Pre = \frac{TP}{TP + FP} \tag{5}$$

$$Rec = \frac{TP}{TP + FN} \tag{6}$$

$$FAR = \frac{FP}{TN + FP} \tag{7}$$

5.4. Performance results and analysis of seen and unseen attacks detection

This subsection summarizes the results of our proposed IDS in detecting seen and unseen attacks and provides an analysis of these findings.

Starting with the results for seen attack detection in the first model, the ANN was trained and tested on a labeled dataset that included normal data and four types of attacks: DoS, frame fuzzification, RPM, and gear spoofing. The performance of the ANN model is detailed in Table 9, which shows that the ANN model consistently achieved impressive accuracy and F1-scores exceeding 99% in accurately classifying various types of seen attacks. Additionally, the table highlights the model's precision, its DR, and FAR for normal data and each attack category, demonstrating the model's high reliability and effectiveness in distinguishing between normal and various types of attacks. Fig. 8 displays the multiclass confusion matrix (CM) of the ANN model, which illustrates the model's capability to classify test data into multiple attack categories. Moreover, as depicted in Fig. 9, the training and validation losses decreased and converged over time, which indicates that the model was learning effectively and generalizing well without overfitting.

Table 9
Performance Evaluation of ANN on Seen Attacks Detection

Attack	Acc (%)	F1	Pre	DR (%)	FAR (%)
DoS	99.99	1.00	1.00	100	0.0
Frame Fuzzification	99.99	0.99	0.99	99.95	0.0005
Gear	99.99	1.00	0.100	100	0.0
RPM	99.99	0.99	0.99	100	0.0
Normal	99.99	0.99	0.99	99.99	0.012

To evaluate the model's ability to detect new, unseen attacks, we trained the LSTM-autoencoder on a sample of normal data and then tested it on the remaining dataset. As shown in Table 10, the LSTM-autoencoder performed well, with an overall accuracy of 98.59%, an F1-score of 0.95, a DR of 99.99%, and a precision of 0.91 across all types of unseen attacks. Despite generating approximately 0.016% FAR, these metrics underscore the model's efficacy in detecting new, unseen at-

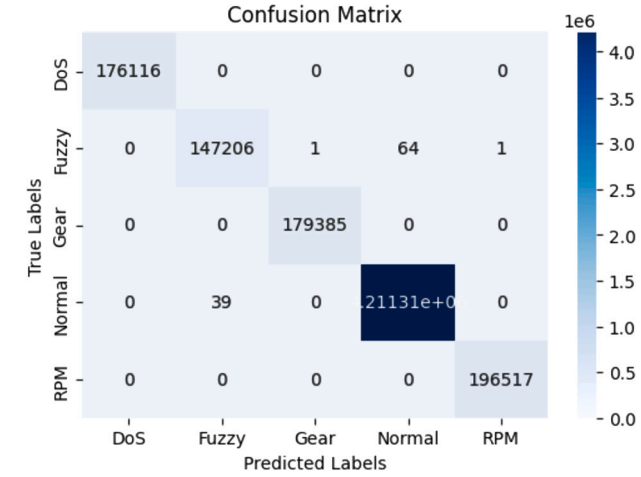


Fig. 8. ANN Multiclass Confusion Matrix

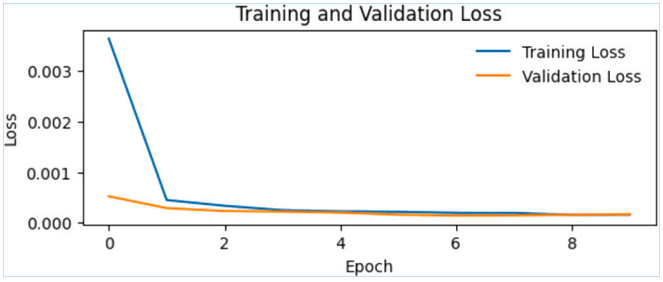


Fig. 9. ANN Training and validation performance

Table 10
Detection Results for Unseen Attacks

Unseen Attack	F1	Pre	Acc (%)	DR (%)	FAR (%)
DoS	0.83	0.71	98.42	100	0.016
Frame Fuzzification	0.81	0.68	98.41	99.99	0.016
Gear	0.84	0.72	98.42	100	0.016
RPM	0.85	0.74	98.42	100	0.016
All	0.95	0.91	98.59	99.99	0.016

tacks. For each type of unseen attack, the model successfully detected all attacks with a DR between 99.99% and 100%, accuracy exceeding 98%, and low FAR of 0.016%. However, the F1-score varied from 0.81 to 0.85 for specific types of unseen attacks. Fig. 10 shows the binary CM of the LSTM-autoencoder model, illustrating its performance in classifying the test data into normal (0) and anomaly (1) classes. Fig. 10a shows the CM on the test samples after applying SMOT sampling, as described in Section 4.1.1, while Fig. 10b displays the CM for the remaining test set in the dataset. From the results, it is clear that the model successfully detected all unseen attacks, even when it had not been trained on them before.

Our results indicate that the ANN successfully detected and classified attacks by type. This capability is crucial, as identifying the specific attack type aids in selecting appropriate countermeasures and conducting post-attack analysis [65]. As shown in Table 10, the model showed different F1-scores for unseen attack detection when testing each attack individually compared to testing all attacks together. The F1-score varied from approximately 0.81 to 0.95. This is because the number of TPs significantly increased when combined, while the number of FPs remained similar. Consequently, this increase in TPs improved both precision and recall, resulting in a higher F1-score. Moreover, from the high DR and the constant FAR across all results, we can observe that while the model was able to detect all unseen attacks, it mistakenly classified some normal data as attacks, accounting for 234,994 FPs. One reason for this could be that the model was trained on a small sample of normal data. Although having 234,994 FPs out of approximately 14,000,000 is a good result, it should be further improved upon in such a critical application. Nevertheless, our findings reveal promising results in detecting unseen attacks.

5.5. Model complexity

This section discusses model complexity in terms of model size (in megabytes, MB) and the number of trainable parameters. When designing in-vehicle IDS solutions, it is essential to consider the deployment requirements [66]. The development and deployment of IDSs are significantly impacted by the constraints of ECU in-vehicle networks, which include limited memory storage, computing power, and bandwidth [15]. In pursuit of optimal results, we have simplified the model architecture to minimize its size. The proposed IDS achieves this reduction by employing a straightforward architecture with a minimal number of layers and neurons in both models, as well as the dropout regularization technique in the LSTM-autoencoder. Through careful experimentation with hyperparameters, we optimized the model's efficiency, resulting in a lightweight architecture. The sizes of the ANN and LSTM-autoencoder models were calculated to be 0.030 MB and 2.95 MB, respectively. Moreover, the number of parameters significantly influences the model's training and testing time. In theory, a model with fewer parameters will train and test more quickly [14]. The ANN model has 517 trainable parameters, while the LSTM-autoencoder has 253,065, making a combined total of 253,582 trainable parameters.

5.6. Comparison with existing studies

This model is compared with recent work in [25], since they used the same dataset and a similar approach and features. Regarding the seen attack detection results, both have a high DR with an F1-score of 0.99. However, in detecting unseen attacks, even though it is difficult to obtain a fair comparison, we made an effort to make the best possible comparison. To do so, we used the same numbers of testing instances for attack and normal instances as were used in [25]. Results in Table 11 show that our model outperformed the results in [25], with a higher DR and lower FAR. For the F1-score, our average was 0.95, while their model achieved a slightly higher score of 0.96. However, the F1-score for unseen attacks in [25] was initially around 0.83, and they improved the result to 0.96 by implementing two biased classifiers after the unsupervised model, achieving a DR of around 93%. Training these biased classifiers on FPs and FNs, however, transforms the model from being purely unsupervised. Although [14] and [24] used the same dataset as ours, we did not compare our detection results with theirs because they relied solely on the CAN ID feature to build their models.

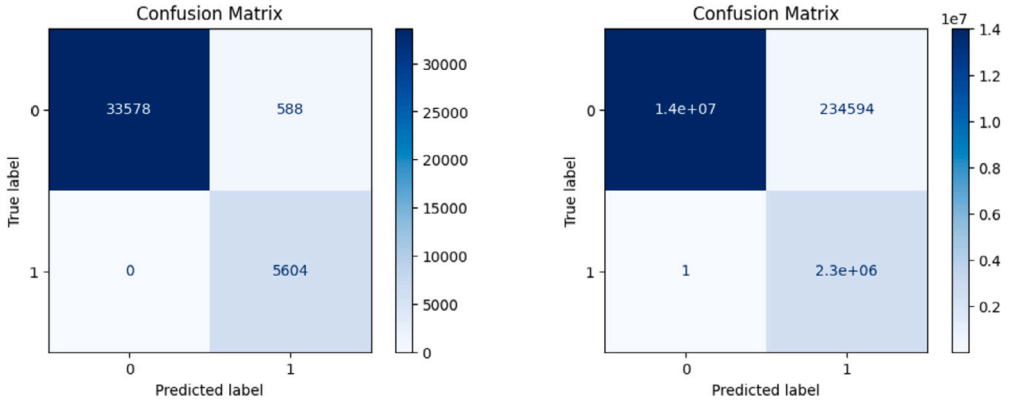
Most previous papers do not state the model sizes, except [25] and [14], so we compared our model with theirs. As depicted in Table 12 the model size in [25] for the two models is 2.61 MB, and the total size of our models is 2.98 MB, showing that our model is nearly in the same range even though we used DL, which is considered more resource-intensive than traditional ML. These sizes are notably below the typical memory capacity of vehicle-level machines, which can exceed 1 GB of RAM [25]. Moreover, Table 12 shows that our trainable parameters represent approximately an 88.2% reduction compared to the number of trainable parameters in [14], even though they only used one feature, which is the CAN ID.

Therefore, the experimental results confirm that our proposed DL-based, in-vehicle IDS is highly efficient and can effectively detect various types of seen and unseen cyberattacks. Additionally, its lightweight design makes it feasible for real-world deployment.

6. Discussion

Our analysis revealed several key findings that contribute to the understanding and development of in-vehicle IDSs:

- Hybrid IDSs, such as our proposed IDS, can be a robust solution that not only addresses current threats but also prepares for future



(a) SMOT Sampeling (b) All Remaining Testing Data

Fig. 10. LSTM-autoencoder Binary Confusion Matrices

Table 11
Comparison with Existing Work

Unseen Attack	Ours			MTHIDS [25]		
	DR (%)	FAR (%)	F1	DR (%)	FAR (%)	F1
DoS	100	0.016	0.95	100	0.0	1.0
Frame Fuzzification	100	0.016	0.94	73	0.057	0.84
Gear	100	0.016	0.95	100	0.45	0.99
RPM	100	0.016	0.95	100	0.003	0.99
Average	100	0.016	0.95	93.7	0.128	0.96

Table 12
Model Size comparison

Model	MB	Trainable parameters
MTHIDS [25]	2.61	-
AE- GAN [14]	-	2.15 million
Ours	2.98	253,582

ones. Moreover, the order of each approach is important. For example, we adopted the seen attack detection before anomaly detection approach for two reasons: to quickly detect any seen attacks and to double-check the normal data in case an attack bypasses the first model.

- When designing an in-vehicle IDS, several critical decisions should be made during the design phase. One such decision is to include both CAN ID and payload data without feature selection, as attackers might exploit any neglected features in the future [30,31].
- Our analysis of CAN bus data shows that each CAN ID has unique data patterns, so important features for one ID may not be relevant for another. This variability makes it impractical to select a consistent set of important features for all CAN IDs, indicating the need to customize feature selection to improve model performance and generalization.
- The most important finding is that our results prove that DL algorithms can improve the performance of an IDS while meeting the model size requirements in resource-constrained environments.
- Theoretically, and based on the literature review, an H-FL architecture, which adds an edge layer between the central server and the vehicles, can overcome several challenges in the traditional FL architecture that consists only of a server and vehicles. Fig. 11 depicts the theoretical framework of the proposed H-FL.

7. Conclusions and future directions

The aim of this paper was to propose a robust and lightweight multi-stage IDS designed for in-vehicle network security that is capable of detecting both seen and novel attacks. Our IDS addresses the limitations of existing solutions by utilizing a hybrid approach and advanced DL algorithms. To further enhance our IDS and leverage diverse driving behaviors while preserving data privacy, we have proposed a theoretical framework for deploying our IDS in an H-FL environment. We evaluated the performance of our IDS using a real-world dataset containing various cyberattacks, including DoS, frame fuzzification, RPM, and gear spoofing. Experimental results demonstrate that the ANN model effectively classifies seen attacks with an outstanding F1-score of 0.99. Simultaneously, the LSTM-autoencoder model excels at detecting novel attacks, achieving an F1-score of over 0.95 and a DR of 99.99% with minimal false alarms. Overall, our proposed IDS effectively detects both seen and novel attacks within in-vehicle networks and continually updates its knowledge by identifying new, previously unseen attacks, ensuring ongoing improvement over time. Additionally, our IDS is designed to be lightweight, making it suitable for real-world deployment. By detecting both seen and novel attacks, our IDS not only addresses current threats but also prepares for future ones. For future work, we plan to deploy our proposed IDS in a realistic H-FL environment and evaluate its performance.

Although our proposed IDS shows promising results in detecting both seen and novel attacks while maintaining a compact model size, it has certain limitations. Our IDS has been trained and evaluated within limited driving scenarios, requiring extensive datasets to model normal behavior accurately. This limitation suggests potential areas for enhancement, which can be addressed through the following future directions:

- Future research could explore streaming learning, allowing the model to dynamically adjust in real time within the vehicle to adapt to various driving conditions, thus enhancing detection accuracy.
- FL can effectively combine models derived from different driving scenarios and vehicle states, greatly improving in-vehicle IDS performance while protecting data privacy and reducing latency [40]. Thus, exploring the field of FL and addressing its challenges, such as data heterogeneity [67] and secure communication [68], can be identified as future trends in in-vehicle IDS research.
- Another crucial future direction is protecting in-vehicle IDSs from adversarial attacks, as a recent study [69] has highlighted their vulnerability. Protecting in-vehicle IDSs from adversarial attacks and adapting solutions from other domains could provide valuable insights and improvements to current in-vehicle IDSs.

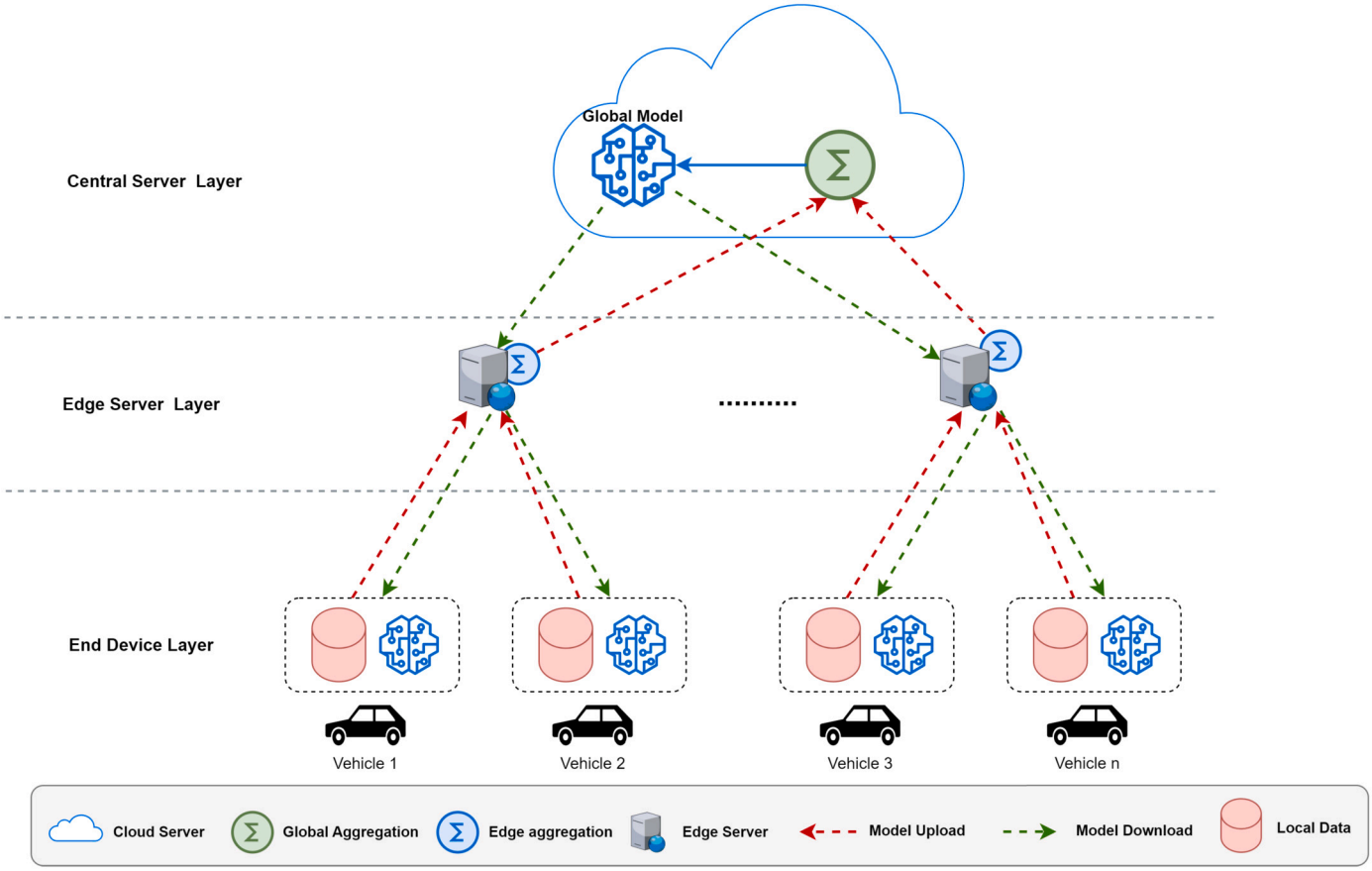


Fig. 11. Framework of the proposed Hierarchical Federated Learning Method

CRediT authorship contribution statement

Muzun Althunayyan: Conceptualization, Formal analysis, Methodology, Resources, Validation, Writing – original draft, Writing – review & editing. **Amir Javed:** Conceptualization, Resources, Supervision, Writing – review & editing. **Omer Rana:** Resources, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used ChatGPT 3.5/4 in order to improve readability and language. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

[1] I. Foster, K. Koscher, Exploring controller area networks, login, USENIX Assoc. 40 (6) (2015).
[2] O.Y. Al-Jarrah, C. Maple, M. Dianati, D. Oxtoby, A. rMouzakitis, Intrusion detection systems for intra-vehicle networks: a review, IEEE Access 7 (2019) 21266–21289, https://doi.org/10.1109/ACCESS.2019.2894183.
[3] A. Paul, M.R. Islam, An artificial neural network based anomaly detection method in can bus messages in vehicles, in: 2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI), IEEE, 2021, pp. 1–5.
[4] M.D. Pesé, J.W. Schauer, J. Li, K.G. Shin, S2-can: sufficiently secure controller area network, in: Proceedings of the 37th Annual Computer Security Applications Conference, 2021, pp. 425–438.
[5] A. Barati, A. Movaghar, M. Sabaei, Energy efficient and high speed error control scheme for real time wireless sensor networks, Int. J. Distrib. Sens. Netw. 10 (5) (2014) 698125, https://doi.org/10.1155/2014/698125.
[6] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, Comprehensive experimental analyses of automotive attack surfaces, in: 20th USENIX Security Symposium (USENIX Security 11), USENIX Association, San Francisco, CA, 2011.
[7] C. Young, J. Zambreno, H. Olufowobi, G. Bloom, Survey of automotive controller area network intrusion detection systems, IEEE Des. Test 36 (6) (2019) 48–55, https://doi.org/10.1109/MDAT.2019.2899062.
[8] J. Golson, Jeep hackers at it again, this time taking control of steering and braking systems, https://www.theverge.com/2016/8/2/12353186/car-hack-jeep-cherokee-vulnerability-miller-valasek. (Accessed 1 April 2023), 2016.
[9] J. Crume, Ownstar: yet another car hack, https://insideinternetsecurity.wordpress.com/2015/08/05/ownstar-yet-another-car-hack/. (Accessed 20 March 2023), 2015.
[10] TKS Lab, New vehicle security research by keenlab: experimental security assessment of bmw cars, https://keenlab.tencent.com/en/2018/05/22/New-CarHacking-Research-by-KeenLab-Experimental-Security-Assessment-of-BMW-Cars/. (Accessed 10 April 2023), 2018.
[11] TKS Lab, Experimental security assessment on lexus cars, https://keenlab.tencent.com/en/2020/03/30/Tencent-Keen-Security-Lab-Experimental-Security-Assessment-on-Lexus-Cars/. (Accessed 10 April 2023), 2020.
[12] M. Bertoncello, C. Martens, T. Möller, T. Schneiderbauer, Unlocking the full life-cycle value from connected-car data, https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/unlocking-the-full-life-cycle-value-from-connected-car-data. (Accessed 6 April 2023), 2021.
[13] Robert Bosch GmbH, Postfach 50, 1991, p. 15.
[14] T.-N. Hoang, D. Kim, Detecting in-vehicle intrusion via semi-supervised learning-based convolutional adversarial autoencoders, Veh. Commun. 38 (2022), https://doi.org/10.1016/j.vehcom.2022.100520.
[15] S. Rajapaksha, H. Kalutarage, M.O. Al-Kadri, A. Petrovski, G. Madzudzo, M. Cheah, Ai-based intrusion detection systems for in-vehicle networks: a survey, ACM Comput. Surv. 55 (11) (2023) 1–40, https://doi.org/10.1145/3570954.

[16] E. Aliwa, O. Rana, C. Perera, P. Burnap, Cyberattacks and countermeasures for in-vehicle networks, ACM Comput. Surv. 54 (1) (2021) 1–37, https://doi.org/10.1145/3431233.
[17] H. Lee, S.H. Jeong, H.K. Kim, Otids: a novel intrusion detection system for in-vehicle network by using remote frame, in: 2017 15th Annual Conference on Privacy, Security and Trust (PST), IEEE, 2017, pp. 57–5709.
[18] T. Hoppe, S. Kiltz, J. Dittmann, Applying intrusion detection to automotive it-early insights and remaining challenges, J. Inf. Assur. Secur. 4 (6) (2009) 226–235.
[19] M.D. Hossain, H. Inoue, H. Ochiai, D. Fall, Y. Kadobayashi, Lstm-based intrusion detection system for in-vehicle can bus communications, IEEE Access 8 (2020) 185489–185502, https://doi.org/10.1109/ACCESS.2020.3029307.
[20] M.D. Hossain, H. Inoue, H. Ochiai, D. Fall, Y. Kadobayashi, An effective in-vehicle can bus intrusion detection system using cnn deep learning approach, in: GLOBECOM 2020-2020 IEEE Global Communications Conference, IEEE, 2020, pp. 1–6.
[21] H.M. Song, H.K. Kim, Self-supervised anomaly detection for in-vehicle network using noised pseudo normal data, IEEE Trans. Veh. Technol. 70 (2) (2021) 1098–1108, https://doi.org/10.1109/TVT.2021.3051026.
[22] P. Wei, B. Wang, X. Dai, L. Li, F. He, A novel intrusion detection model for the can bus packet of in-vehicle network based on attention mechanism and autoencoder, Digit. Commun. Netw. (2022), https://doi.org/10.1016/j.dcan.2022.04.021.
[23] J. Zhang, F. Li, H. Zhang, R. Li, Y. Li, Intrusion detection system using deep learning for in-vehicle security, Ad Hoc Netw. 95 (2019) 101974, https://doi.org/10.1016/j.adhoc.2019.101974.
[24] E. Seo, H.M. Song, H.K. Kim, Gids: gan based intrusion detection system for in-vehicle network, in: 2018 16th Annual Conference on Privacy, Security and Trust (PST), IEEE, 2018, pp. 1–6.
[25] L. Yang, A. Moubayed, A. Shami, Mth-ids: a multitiered hybrid intrusion detection system for Internet of vehicles, IEEE Int. Things J. 9 (1) (2022) 616–632, https://doi.org/10.1109/JIOT.2021.3084796.
[26] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization, in: ICISSp 1, 2018, pp. 108–116.
[27] A. Vikram, et al., Anomaly detection in network traffic using unsupervised machine learning approach, in: 2020 5th International Conference on Communication and Electronics Systems (ICCES), IEEE, 2020, pp. 476–479.
[28] B.A. Pratomo, P. Burnap, G. Theodorakopoulos, Unsupervised approach for detecting low rate attacks on network traffic with autoencoder, in: 2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), IEEE, 2018, pp. 1–8.
[29] G. Kocher, G. Kumar, Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges, Soft Comput. 25 (15) (2021) 9731–9763, https://doi.org/10.1007/s00500-021-05893-0.
[30] B. Li, Y. Vorobeychik, Feature cross-substitution in adversarial classification, Adv. Neural Inf. Process. Syst. 27 (2014).
[31] F. Zhang, P.P. Chan, B. Biggio, D.S. Yeung, F. Roli, Adversarial feature selection against evasion attacks, IEEE Trans. Cybern. 46 (3) (2015) 766–777, https://doi.org/10.1109/TCYB.2015.2415032.
[32] S.T. Mehedi, A. Anwar, Z. Rahman, K. Ahmed, Deep transfer learning based intrusion detection system for electric vehicular networks, Sensors 21 (14) (2021) 4736, https://doi.org/10.3390/s21144736.
[33] J. Nagarajan, P. Mansourian, M.A. Shahid, A. Jaekel, I. Saini, N. Zhang, M. Kneppers, Machine learning based intrusion detection systems for connected autonomous vehicles: a survey, Peer-to-Peer Netw. Appl. (2023) 1–33, https://doi.org/10.1007/s12083-023-01508-7.
[34] B. Lampe, W. Meng, A survey of deep learning-based intrusion detection in automotive applications, Expert Syst. Appl. 221 (2023) 119771, https://doi.org/10.1016/j.eswa.2023.119771.
[35] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P.K.R. Maddikunta, T.R. Gadekallu, Federated learning for intrusion detection system: concepts, challenges and future directions, Comput. Commun. 195 (2022) 346–361, https://doi.org/10.1016/j.comcom.2022.09.012.
[36] J. Alsamiri, K. Alsubhi, Federated learning for intrusion detection systems in internet of vehicles: a general taxonomy, applications, and future directions, Future Internet 15 (12) (2023) 403, https://doi.org/10.3390/fi15120403.
[37] M. Driss, I. Almomani, Z. e Huma, J. Ahmad, A federated learning framework for cyberattack detection in vehicular sensor networks, Complex Intell. Syst. 8 (5) (2022) 4221–4235, https://doi.org/10.1007/s40747-022-00705-w.
[38] K.H. Shibly, M.D. Hossain, H. Inoue, Y. Taenaka, Y. Kadobayashi, Personalized federated learning for automotive intrusion detection systems, in: 2022 IEEE Future Networks World Forum (FNWF), IEEE, 2022, pp. 544–549.
[39] T. Yu, G. Hua, H. Wang, J. Yang, J. Hu, Federated-lstm based network intrusion detection method for intelligent connected vehicles, in: IEEE International Conference on Communications (ICC), 2022.
[40] H. Zhang, K. Zeng, S. Lin, Federated graph neural network for fast anomaly detection in controller area networks, IEEE Trans. Inf. Forensics Secur. 18 (2023) 1566–1579, https://doi.org/10.1109/TIFS.2023.3240291.
[41] J. Yang, J. Hu, T. Yu, Federated AI-enabled in-vehicle network intrusion detection for internet of vehicles, Electronics (2022), https://doi.org/electronics11223658.
[42] O. Rana, T. Spyridopoulos, N. Hudson, M. Baughman, K. Chard, I. Foster, A. Khan, Hierarchical and decentralised federated learning, in: 2022 Cloud Continuum, IEEE, 2022, pp. 1–9.
[43] L. Liu, J. Zhang, S. Song, K.B. Letaief, Client-edge-cloud hierarchical federated learning, in: ICC 2020-2020 IEEE International Conference on Communications (ICC), IEEE, 2020, pp. 1–6.
[44] K.M. Almatar, Traffic congestion patterns in the urban road network:(dammam metropolitan area), Ain Shams Eng. J. 14 (3) (2023) 101886, https://doi.org/10.1016/j.asej.2022.101886.
[45] K.M. Almatar, Increasing Electric Vehicles Infrastructure in Urban Areas for Efficiently Employing Renewable Energy, Environment, Development and Sustainability, 2023, pp. 1–22.
[46] K.M. Almatar, Towards sustainable green mobility in the future of Saudi Arabia cities: implication for reducing carbon emissions and increasing renewable energy capacity, Heliyon 9 (3) (2023), https://doi.org/10.1016/j.heliyon.2023.e13977.
[47] M.M. Rahman, J.-C. Thill, Impacts of connected and autonomous vehicles on urban transportation and environment: a comprehensive review, Sustain. Cities Soc. 96 (2023) 104649, https://doi.org/10.1016/j.scs.2023.104649.
[48] U.S. Musa, M. Chhabra, A. Ali, M. Kaur, Intrusion detection system using machine learning techniques: a review, in: 2020 International Conference on Smart Electronics and Communication (ICOSEC), IEEE, 2020, pp. 149–155.
[49] A. Hbaieb, S. Ayed, L. Chaari, Federated learning based ids approach for the iov, in: Proceedings of the 17th International Conference on Availability, Reliability and Security, 2022, pp. 1–6.
[50] K. Faraoun, A. Boukelif, Neural networks learning improvement using the k-means clustering algorithm to detect network intrusions, INFOCOMP J. Comput. Sci. 5 (3) (2006) 28–36.
[51] K.M. Ali Alheeti, K. McDonald-Maier, Intelligent intrusion detection in external communication systems for autonomous vehicles, Syst. Sci. Control Eng. 6 (1) (2018) 48–56, https://doi.org/10.1080/21642583.2018.1440260.
[52] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (1943) 115–133, https://doi.org/10.1007/BF02478259.
[53] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, Psychol. Rev. 65 (6) (1958) 386, https://doi.org/10.1037/h0042519.
[54] J.V. Tu, Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes, J. Clin. Epidemiol. 49 (11) (1996) 1225–1231, https://doi.org/10.1016/S0895-4356(96)00002-9.
[55] G.P. Zhang, Neural networks for classification: a survey, IEEE Trans. Syst. Man Cybern., Part C, Appl. Rev. 30 (4) (2000) 451–462, https://doi.org/10.1109/5326.897072.
[56] S. Dreiseitl, L. Ohno-Machado, Logistic regression and artificial neural network classification models: a methodology review, J. Biomed. Inform. 35 (5–6) (2002) 352–359, https://doi.org/10.1016/S1532-0464(03)00034-0.
[57] J. Wu, D. Peng, Z. Li, L. Zhao, H. Ling, Network intrusion detection based on a general regression neural network optimized by an improved artificial immune algorithm, PLoS ONE 10 (3) (2015) e0120976, https://doi.org/10.1371/journal.pone.0120976.
[58] A. Shenfield, D. Day, A. Ayesh, Intelligent intrusion detection systems using artificial neural networks, ICT Express 4 (2) (2018) 95–99, https://doi.org/10.1016/j.icte.2018.04.003.
[59] V.K. Kukkala, S.V. Thiruloga, S. Pasricha, Indra: intrusion detection using recurrent autoencoders in automotive embedded systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 39 (11) (2020) 3698–3710, https://doi.org/10.1109/TCAD.2020.3012749.
[60] L. Gao, F. Li, X. Xu, Y. Liu, Intrusion detection system using soeks and deep learning for in-vehicle security, Clust. Comput. 22 (6) (2019) 14721–14729, https://doi.org/10.1007/s10586-018-2385-7.
[61] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.
[62] L. Li, Y. Fan, M. Tse, K.-Y. Lin, A review of applications in federated learning, Comput. Ind. Eng. 149 (2020) 106854, https://doi.org/10.1016/j.cie.2020.106854.
[63] E.M. Campos, P.F. Saura, A. González-Vidal, J.L. Hernández-Ramos, J.B. Bernabé, G. Baldini, A. Skarmeta, Evaluating federated learning for intrusion detection in internet of things: review and challenges, Comput. Netw. 203 (2022) 108661, https://doi.org/10.1016/j.comnet.2021.108661.
[64] X.1375 Working Group, Guidelines for an intrusion detection system for in-vehicle networks, Tech. Rep. X.1375, International Telecommunication Union, 2020.
[65] Q. Zhao, M. Chen, Z. Gu, S. Luan, H. Zeng, S. Chakrabory, Can bus intrusion detection based on auxiliary classifier gan and out-of-distribution detection, ACM Trans. Embed. Comput. Syst. 21 (4) (2022) 1–30, https://doi.org/10.1145/3540198.
[66] S.-F. Lokman, A.T. Othman, M.-H. Abu-Bakar, Intrusion detection system for automotive controller area network (can) bus system: a review, EURASIP J. Wirel. Commun. Netw. 2019 (2019) 1–17, https://doi.org/10.1186/s13638-019-1484-3.
[67] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, Proc. Mach. Learn. Syst. 2 (2020) 429–450.
[68] A. Barati, A. Movaghar, M. Sabaei, Rdtp: reliable data transport protocol in wireless sensor networks, Telecommun. Syst. 62 (2016) 611–623, https://doi.org/10.1007/s11235-015-0098-2.
[69] F. Aloraini, A. Javed, O. Rana, Adversarial attacks on intrusion detection systems in in-vehicle networks of connected and autonomous vehicles, Sensors 24 (12) (2024) 3848, https://doi.org/10.3390/s24123848.