Detection of Cyberattacks in In-Vehicle Networks Using a Hierarchical Federated Learning Framework

Muzun Saleh Althunayyan

A thesis submitted for the degree of
Doctor of Philosophy

Cardiff University

September 2025

Abstract

Connected and Autonomous Vehicles (CAVs) are expected to become the backbone of future transportation systems. However, the primary protocol for in-vehicle communication, the Controller Area Network (CAN) bus, was not designed with security in mind. As CAVs become increasingly interconnected, this lack of built-in security exposes them to a range of cyberattacks, posing significant risks, including the potential loss of human life. Machine Learning (ML)-based Intrusion Detection Systems (IDSs) have proven effective in identifying cyberattacks on in-vehicle networks. However, existing studies often overlook three critical requirements for in-vehicle IDSs: robustness, resource constraints, and the deployment environment.

The overarching aim of this thesis is to improve the security of in-vehicle networks by first analysing CAN bus data to assess the feasibility and effectiveness of feature selection techniques for building a robust IDS. Based on these findings, we propose a novel, multi-stage IDS designed to detect and classify known attacks while also identifying previously unseen attacks. Although the proposed IDS leverages Deep Learning (DL) algorithms, it remains lightweight with a low memory footprint.

Most existing approaches rely on centralised training for deployment, which involves transmitting raw data to a central server, raising significant privacy concerns. Federated Learning (FL) addresses these issues by enabling local model training and transmitting model updates instead of raw data. To further enhance IDS robustness, preserve data privacy, and overcome the single point of failure in standard FL, the proposed IDS is deployed within a simulated Hierarchical Federated Learning (H-FL) framework. This framework incorporates multiple edge servers and a central aggregator, enabling collaborative learning from diverse data sources and improving the detection of unknown attacks beyond the coverage of any single server.

Table of contents

List of Tables

List of Figures

List of Abbreviations

ACK Acknowledge Field

AE Autoencoder

ANN Artificial Neural Network

BCNN Binarized CNN

Bi-LSTM Bidirectional LSTM

BNN Binarised Neural Network

CAN Controller Area Network

CAV Connected and Autonomous Vehicle

CNN Convolutional Neural Network

CRC Cyclic Redundancy Check

DBC DataBase CAN

DCNN Deep Convolutional Neural Network

DL Deep Learning

DLC Data Length Code

DR Detection Rate

DNN Deep Neural Network

DoS Denial-of-Service

DTC Decision Tree Classifier

DTs Decision Trees

EOF End of Frame

ET Extra Trees

FAR False Alarm Rate

FL Federated Learning

FN False Negative

FP False Positive

FPGA Field Programmable Gate Array

GAN Generative Adversarial Network

GDM Gradient Descent with Momentum

GMM Gaussian Mixture Model

GNB Gaussian Naïve Bayes

GP Genetic Programming

GRU Gated Recurrent Units

H-FL Hierarchical Federated Learning

ID Arbitration Field

IDS Intrusion Detection System

IL Incremental Learning

IoV Internet of Vehicles

IRC Internet Relay Chat

KNN k-nearest Neighbours

KNC k-Nearest Neighbors Classifier

LDAC Linear Discriminant Analysis Classifier

LIN Local Interconnect Network

LR Logistic Regression

LRC Logistic Regression Classifier

LSTM Long and Short Term Memory

MBA Modified Bat Algorithm

MGA Modified Genetic Algorithm

ML Machine Learning

MLP Multi-Layer Perceptron

MOST Media Oriented System Transport

NB Naive Bayes

Non-IID Non-Independent and Identically Distributed

OBD-II On-board Diagnostic II Port

OEM Original Equipment Manufacturer

OCSVM One-Class Support Vector Machine

RF Random Forest

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

SMMT Society of Motor Manufacturers and Traders

SMOTE Synthetic Minority Oversampling Technique

SOF Start of Frame

SOME Same Origin Method Execution

SVM Support Vector Machine

SVC Support Vector Classifier

TCNA Temporal Convolutional Neural Attention

TP True Positive

TN True Negative

WMA Windows Media Audio

XGBoost Extreme Gradient Boosting

List of Publications

- **Paper 1 [Conference paper]** Althunayyan, Muzun, Amir Javed, and Omer Rana. "An innovative feature selection approach for CAN bus data leveraging constant value analysis." International Conference on Cyber Security, Privacy in Communication Networks. Singapore: Springer Nature Singapore, 2023.
- **Paper 2 [Journal paper]** Althunayyan, Muzun, Amir Javed, and Omer Rana. "A robust multi-stage intrusion detection system for in-vehicle network security using hierarchical federated learning." Vehicular Communications 49 (2024): 100837.
- **Paper 3 [Journal paper]** Althunayyan, Muzun, Amir Javed, Omer Rana, and Theodoros Spyridopoulos. "Hierarchical federated learning-based intrusion detection for in-vehicle networks." Future Internet 16, no. 12 (2024): 451.
- **Paper 4 [Journal paper]** Althunayyan, Muzun, Amir Javed, and Omer Rana. "A Survey of Learning-Based Intrusion Detection Systems for In-Vehicle Networks." Computer Networks (2026): 112031.

Dedication

I dedicate this thesis to my home country and beloved family, for their constant love, unwavering support, and heartfelt encouragement.

Acknowledgements

I would like to take this opportunity to extend my heartfelt gratitude to the individuals whose invaluable contributions have shaped the course of my doctoral journey and made this research possible. Their unwavering support, expertise, and encouragement have been instrumental throughout this process.

First and foremost, I would like to praise and thank God (Allah) Almighty for granting me the strength, health, and patience to complete this journey.

I would like to express my deepest appreciation to my primary supervisor, Dr. Amir Javed, for his guidance, support, and incredible patience. His kindness and understanding, especially during moments of stress and uncertainty, provided me with the reassurance I needed to keep moving forward. His ability to guide me through challenges while offering encouragement and clarity has been a cornerstone of my journey, and I am truly grateful for his compassionate mentorship.

I am equally indebted to my second supervisor, Professor Omer Rana, whose wealth of knowledge and generosity in sharing his insights has been a source of constant inspiration.

Additionally, this endeavor would not have been possible without the generous support from the Royal Embassy of Saudi Arabia Cultural Bureau and Majmaah University. Their financial sponsorship throughout these years enabled me to fully dedicate myself to this research.

To my incredible colleagues, friends, and office mates, words cannot express how much your support has meant to me. Your encouragement, shared laughter, and thoughtful conversations carried me through the most challenging times. You have not only been collaborators but a true source of strength. Special thanks also to Dr. Theodoros Spyridopoulos for his invaluable contributions to the federated learning aspects of this research.

I would like to express my deepest appreciation to my best friend, Fatimah Aloraini, who provided distractions when needed and encouragement when it seemed impossible to continue. Your friendship has truly been a blessing throughout this journey.

Most importantly, this thesis is dedicated to my parents and my sisters. Their unwavering love, belief in me, and constant encouragement have been the foundation of everything I have accomplished. This achievement is as much theirs as it is mine.

This thesis stands as a testament to the collaborative efforts of many. I am humbled and deeply grateful for the guidance and support that have shaped my academic journey.

Chapter 1

Introduction

1.1 Overview

Connected and Autonomous Vehicles (CAVs) are expected to exceed 14 million units in annual production with embedded connectivity by 2025 [3]. As CAVs become more widespread, they are set to become the backbone of future transportation systems [4], offering the potential to not only revolutionise mobility but also deliver significant economic benefits. For example, the Society of Motor Manufacturers and Traders (SMMT) estimates that this technological shift could provide the United Kingdom with an annual economic boost of £62 billion by 2030 [5]. However, these advancements also expand the attack surface of CAVs, rendering them more attractive targets for cyberattacks [6].

CAVs rely on Electronic Control Units (ECUs) to manage and control various functions. These ECUs communicate through standardised in-vehicle communication protocols, such as the Controller Area Network (CAN), FlexRay, Local Interconnect Network (LIN), and Media Oriented System Transport (MOST). Among these, the CAN bus is the protocol that is the most widely used, valued for its high speed, reliability, and ease of use [7]. Although originally designed for industrial applications, the CAN bus has become the de facto standard for in-vehicle communication [8]. Despite its advantages, the CAN protocol was not designed with security in mind and lacks essential features such as sender authentication and encryption [9]. Implementing these security measures is challenging due to the limited computational resources available in vehicles, which could interfere with safety-critical operations [10, 11].

The increasing interconnectivity of CAVs exposes them to a range of cyberattacks. The attack surfaces in modern vehicles can be accessed either physically, via ports such as the USB or the On-Board Diagnostic (OBD)-II port, or remotely through wireless technologies such as Bluetooth, Wi-Fi, and LTE [12]. In 2023, the number of large-scale incidents,

potentially affecting thousands to millions of mobility assets, grew 2.5-fold compared to 2022. Additionally, 95% of cyberattacks are conducted remotely, with 85% being long-range [13]. These vulnerabilities make vehicles susceptible to attacks that could have devastating consequences, including loss of control over critical systems like braking, steering, and acceleration [14]. A recent incident [15] involved cybersecurity researcher Ian Tabor discovering tampering on his Toyota RAV4, particularly around the front bumper and headlight area. Initially suspecting vandalism, Tabor soon realised the vehicle had been targeted by a cyberattack. Investigations revealed that attackers had accessed the car's CAN bus through exposed wiring, allowing them to inject malicious signals. This manipulation enabled the attackers to unlock the doors and start the engine, ultimately stealing the vehicle without the need for a key. Moreover, a notorious example is the Jeep hack, where attackers remotely gained control over the vehicle's braking and steering systems, resulting in dangerous driving conditions [16]. Similarly, vulnerabilities in BMW and Toyota Lexus models have been exploited, demonstrating the persistent threat to vehicle security [17, 18]. Such incidents underscore the need for robust security measures to protect against both information theft and direct physical harm.

Given the severity of these threats, the security of the CAN bus has become a major area of research. According to McKinsey's analysis, by 2030, almost 95% of the new vehicles will be connected to external networks, further highlighting the need for security solutions [19]. One promising approach is the implementation of Intrusion Detection Systems (IDSs), which monitor network traffic for malicious activity. In the context of in-vehicle networks, an IDS is typically deployed on an ECU, where it analyses incoming messages to detect abnormalities. However, conventional IDS technologies designed for traditional networks cannot be applied directly to in-vehicle systems due to resource constraints and the real-time requirements of automotive environments. These constraints make implementing security measures particularly challenging, as they must not interfere with safety-critical operations [10, 11]. Moreover, as attackers increasingly adapt their techniques to evade existing security measures, the frequency of unknown attacks has risen significantly in recent years [20, 21]. If left undetected, such attacks can lead to severe consequences. This underscores the urgent need for an in-vehicle IDS that is both lightweight, making it suitable for practical deployment, and robust, capable of defending against both known and unknown attacks.

The impact of securing in-vehicle networks in CAVs extends beyond the vehicles themselves, influencing the broader field of urban transportation planning, where efficient, safe, and sustainable systems are crucial. Research on mitigating traffic congestion and improving safety in urban environments has shown that CAVs provide substantial benefits, including

reduced energy consumption and lower emissions [22]. They also have the potential to minimise traffic accidents caused by human error while enhancing passenger convenience and productivity [23]. However, these advantages are accompanied by widespread concerns about safety, security, and privacy due to the increasing risk of cyberattacks. Strengthening the security of CAVs is therefore essential not only to protect individual vehicles but also to ensure the resilience and efficiency of urban transportation systems as a whole.

1.2 Motivation and Problem Definition

As modern vehicles integrate advanced connectivity and become increasingly software-driven, the automotive cybersecurity threat landscape has expanded significantly [24]. The CAN protocol, commonly accepted as the primary standard for in-vehicle communication [7], was designed without security features, leading to numerous vulnerabilities that expose vehicles to significant risks. These vulnerabilities, such as the lack of authentication and encryption, are discussed in detail in Chapter 2. Many researchers have exploited these vulnerabilities to carry out cyberattacks [16, 25, 17, 18]. Such attacks can lead to severe consequences, including the potential loss of human life [14]. Although the security community has devoted extensive attention to securing communication protocols, the unique constraints and complexities of embedded systems, along with the CAN protocol, which dates back to the 1980s, make many proposed defences impractical or unfeasible [26]. To protect in-vehicle networks from such cyberattacks, IDSs have proven to be an effective solution [27].

Many studies, detailed in Chapter 3, have developed in-vehicle IDSs utilising various Machine Learning (ML) techniques. Some researchers focus solely on developing IDSs to detect known attacks. Such approaches are highly dependent on the availability of well-labelled attack datasets, whose collection is time-consuming, error-prone, and often impractical. Moreover, by relying on predefined attack patterns, these IDSs have limited ability to handle attack variations or previously unknown attacks. As the frequency of unknown attacks has increased significantly [20], driven by the expansion of connectivity services and attack vectors, other researchers have turned to anomaly-based IDSs to detect unseen attacks. However, these approaches typically lack the ability to assign fine-grained labels that identify specific attack types, which is crucial for selecting timely and appropriate countermeasures and supporting post-attack analysis [28]. Despite these advances, only a limited number of studies have proposed comprehensive in-vehicle IDSs capable of detecting both known and unknown attacks. Existing solutions often overlook memory constraints and real-time requirements, which hinder practical deployment. In addition, they rely on

traditional centralised learning approaches that require transmitting large volumes of data to the cloud, raising concerns related to privacy, communication overhead, and response latency [29]. Therefore, existing approaches fail to address three critical requirements for in-vehicle IDSs: robustness, resource limitations, and deployment environment.

- **Robustness:** It refers to an IDS's ability not only to detect and classify known attacks but also to stay ahead of attackers and identify new, unseen, and adversarial attacks. Strengthening the robustness of an IDS is essential to enhance the safety and security of in-vehicle networks against both current and previously unknown cyberattacks.

- **Resource limitations:** Given the memory constraints of ECUs in in-vehicle networks [30], an efficient IDS must be lightweight and have a small memory footprint [31, 32]. Maintaining low memory overhead is essential to prevent disruption of real-time automotive applications [32, 33] and to enable practical deployment. However, many existing IDSs either rely solely on CAN ID features, use traditional ML methods with manual feature extraction, neglect other potentially exploitable features and compromise security, or fail to consider memory constraints and real-time requirements when designing in-vehicle IDSs [27], making many proposed IDSs impractical for real-world applications.

- **Deployment environment:** Is often overlooked in IDS design. Traditional centralised learning approaches, which involve transmitting large volumes of data to the cloud for training, raise significant concerns, including privacy risks, high communication overhead, and slower response times. Although some studies [34–39] have deployed in-vehicle IDSs in a Federated Learning (FL) environment, they still face limitations. Most FL-based in-vehicle IDSs rely on a standard cloud-based architecture with a single central aggregator, which can limit adaptability, increase network congestion, and cause communication delays when sharing models between the central aggregator and numerous vehicles [40]. Furthermore, this centralised approach introduces a single point of failure, compromising robustness and scalability [41].

Given these limitations, this study aims to address these gaps by proposing a robust and lightweight in-vehicle IDS capable of defending against both known and new, unseen attacks using a Deep Learning (DL) approach, which has demonstrated superior performance in efficiently processing large volumes of data at a faster rate [42]. In contrast to the limitations of traditional FL, our proposed Hierarchical FL (H-FL) framework introduces multiple edge aggregators, which distribute computational loads and reduce latency, addressing both

the scalability and adaptability concerns. By enabling decentralised learning across edge nodes, this architecture allows the IDS to capture a broader range of driving scenarios and adapt to new attacks. It is designed to overcome the limitations of relying solely on a central aggregator [41]. **To the best of our knowledge**, this is the first study to propose a hybrid approach in in-vehicle IDSs that detects both known and unknown attacks using DL algorithms, while maintaining a lightweight design and deploying the IDS within a simulated H-FL framework.

1.3 Scope of the Thesis

This thesis focuses on developing a robust and lightweight in-vehicle IDS capable of detecting both known and unknown attacks using DL algorithms by addressing the limitations of existing in-vehicle IDS solutions. To further enhance the robustness and adaptability of the proposed IDS, this thesis explores the FL approach, allowing the IDS to be deployed locally in each vehicle to preserve data privacy while continually updating and learning from the data of other vehicles. Figure 1.1 demonstrates the scope of this thesis.

1.4 Aim and Objectives

The aim of this thesis is to address the previously identified critical requirements for in-vehicle IDSs by enhancing the security of in-vehicle networks through the development of a robust and lightweight IDS capable of detecting both known and unknown cyberattacks. To achieve this aim, the thesis pursues the following objectives:

1. To survey existing IDSs for in-vehicle networks, with a focus on ML, DL, and FL approaches, in order to identify limitations and research gaps in current solutions.
2. To analyse CAN bus data to gain insights into feature characteristics relevant to the design of a robust in-vehicle IDS.
3. To investigate the reliability of feature selection techniques for improving IDS robustness.
4. To design a lightweight DL-based in-vehicle IDS capable of detecting and classifying known attacks.
5. To extend the IDS to detect previously unseen attacks using anomaly detection techniques.

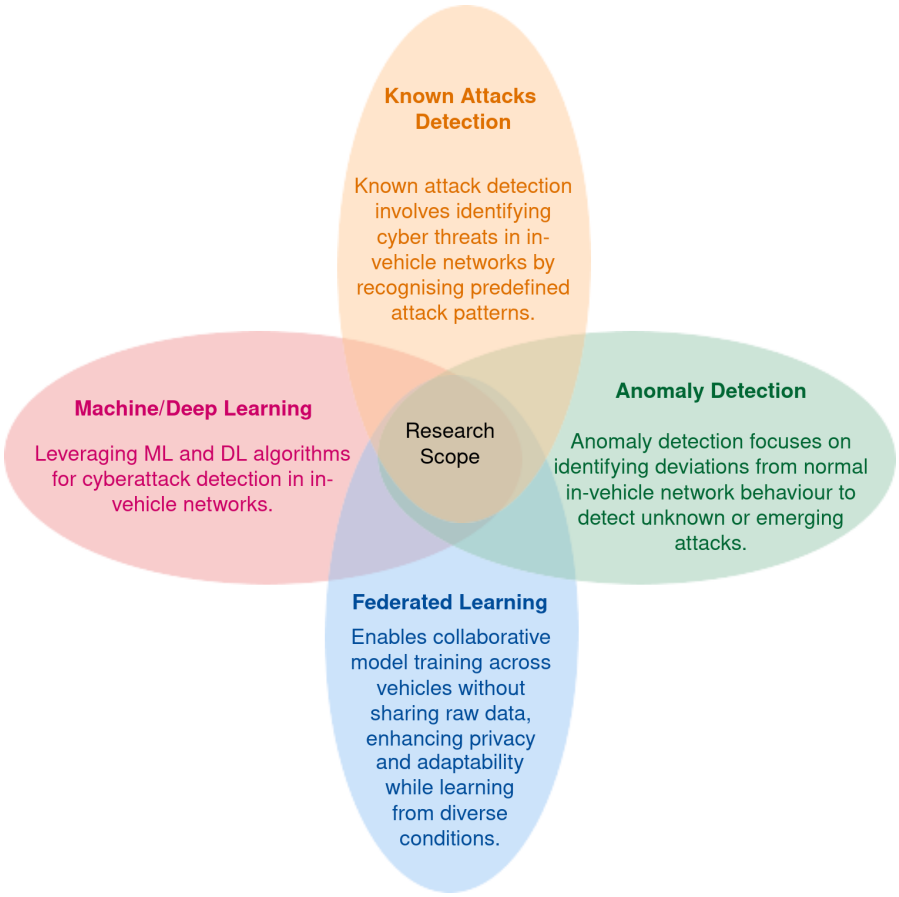


Fig. 1.1 Thesis scope.

6. To evaluate the performance of the proposed IDS using two benchmark datasets.
7. To assess the robustness of the proposed IDS by generating adversarial samples and evaluating their impact on IDS performance.
8. To improve IDS robustness through adversarial training using the generated adversarial samples.
9. To develop a simulated H-FL environment consisting of a central server, edge servers, and clients for IDS deployment.
10. To distribute data across clients in a realistic manner to reflect real-world scenarios.
11. To deploy the proposed IDS within the simulated H-FL environment to address privacy, network congestion, and single point of failure issues associated with standard FL.

1.5 Contributions

The overarching goal of this thesis is to improve the security of in-vehicle networks. This is accomplished through the following key contributions:

C1 This work presents an analysis of CAN bus data that provides insight into its characteristics to support the design of robust IDSs for in-vehicle networks. The analysis shows that each CAN ID exhibits distinct data patterns, meaning that features important for one CAN ID may not be relevant for another. For a given CAN ID, payload values may be zero-constant, constant non-zero, or varying, depending on how the CAN ID is defined in the dictionary. These observations enhance researchers' understanding of CAN bus data and support the development of more robust IDS solutions.

C2 This work evaluates the feasibility and effectiveness of feature selection techniques for building robust IDSs for in-vehicle networks. This evaluation shows that standard feature selection methods often identify payload fields with constant zero values as important while overlooking informative, varying payload fields. Consequently, applying feature selection uniformly across all CAN IDs leads to information loss, as features that are informative for one CAN ID may be zero-constant for others. Overall, these findings clarify how feature selection can adversely affect IDS robustness in in-vehicle networks and provide guidance for more reliable IDS design.

C3 To propose a novel, lightweight in-vehicle IDS that employs DL for efficient and practical deployment. The novelty of the proposed IDS lies in three key aspects: the design of the IDS, the algorithms used, and the deployment approach. The proposed IDS has a total size of 2.98 MB, which is comparable to representative traditional ML–based IDSs with a model size of 2.61 MB, demonstrating that DL can be adopted with minimal additional memory overhead within allowable memory constraints. In addition, the proposed IDS contains 253,582 trainable parameters, which is substantially fewer than those reported in existing DL–based IDS approaches. The lightweight nature of the proposed IDS is achieved through deliberate design choices, including minimising network depth and neuron count while preserving performance, using dropout for regularisation, and adopting a Sequential architecture to avoid unnecessary complexity. Additionally, computationally efficient activation functions are selected to reduce memory usage and inference latency.

C4 To design an IDS capable of detecting and classifying known attacks, including DoS, frame fuzzification, engine RPM spoofing, and drive gear spoofing. These

attacks target manipulation of the CAN ID and/or payload and, when successful, pose significant threats to driver safety. The proposed IDS achieves a high detection rate (DR) with an F1-score of 0.99 for detecting and classifying known attacks. Moreover, it identifies previously unseen attacks with an F1-score above 0.95 and a DR of 99.99%, outperforming comparable studies that report F1-scores of around 0.83 and detection rates of approximately 93%. The IDS also achieves a latency of less than 0.04 ms, meeting the real-time requirements of vehicle safety services. Overall, this provides a robust in-vehicle IDS that classifies known attacks by type, detects unknown attacks, and satisfies the memory and real-time constraints required for in-vehicle deployment, thereby advancing practical IDS design.

C5 To propose and simulate the deployment of an in-vehicle IDS within an H-FL framework using multiple edge servers and a central aggregator, enabling learning from diverse data sources and previously unseen attacks. The H-FL framework mitigates single points of failure found in standard FL, enhances scalability, and efficiently distributes computational load. Experimental results show that deploying the IDS within the H-FL framework improves the F1-score by up to 10.63%, addressing the limitations of edge-based FL in terms of data diversity and attack coverage. Notably, H-FL achieves improved F1-scores in 16 out of 24 evaluated scenarios.

The contributions presented in this thesis have been published as peer-reviewed papers.

Published Papers:

[43] Althunayyan, Muzun, Amir Javed, and Omer Rana. "An innovative feature selection approach for CAN bus data leveraging constant value analysis." International Conference on Cyber Security, Privacy in Communication Networks. Singapore: Springer Nature Singapore, 2023. **[Conference paper]**

[8] Althunayyan, Muzun, Amir Javed, and Omer Rana. "A robust multi-stage intrusion detection system for in-vehicle network security using hierarchical federated learning." Vehicular Communications (2024): 100837. **[Journal paper]**

[44]Althunayyan, Muzun, Amir Javed, Omer Rana, and Theodoros Spyridopoulos. "Hierarchical federated learning-based intrusion detection for in-vehicle networks."

Future Internet 16, no. 12 (2024): 451. **[Journal paper]**

[45] Althunayyan, Muzun, Amir Javed, and Omer Rana. "A Survey of Learning-Based Intrusion Detection Systems for In-Vehicle Networks." Computer Networks (2026): 112031. **[Journal paper]**

Contributions **C1** and **C2** are relevant to [43], **C3** and **C4** are relevant to [8], and **C5** is relevant to [44].

1.6 Thesis Structure

This thesis is structured into seven chapters, with the following content:

Chapter 2 - Background: This chapter provides an overview of in-vehicle protocols, with a focus on the CAN protocol and its relevance to cyberattacks. It explores CAN vulnerabilities, potential entry points to the CAN bus network, and attack scenarios, aiming to offer a deeper understanding of the CAN bus protocol and its potential for exploitation.

Chapter 3 - Literature Review: This chapter provides a comprehensive review of state-of-the-art research on learning-based in-vehicle IDSs, focusing on ML, DL, and FL approaches, aiming to identify limitations and research gaps in existing works.

Chapter 4 - Feature Selection: In this chapter, we provide insight into CAN bus data by analysing its characteristics and evaluating the performance of various feature selection methods across multiple datasets to determine their effectiveness in developing a robust in-vehicle IDS. This chapter presents **C1** and **C2**.

Chapter 5 - Proposed In-vehicle IDS: In this chapter, we propose a novel, lightweight in-vehicle IDS that leverages DL algorithms to overcome the limitations of previous works. The proposed IDS uses a multi-stage approach: an Artificial Neural Network (ANN) in the first stage to detect known attacks, followed by a Long Short-Term Memory (LSTM) autoencoder in the second stage to detect new, unseen attacks. This chapter presents **C3** and **C4**.

Chapter 6 - Hierarchical FL: In this chapter, we evaluate the in-vehicle IDS proposed in Chapter 5 within a simulated H-FL framework, which utilises multiple edge aggregators

and a central aggregator while accounting for realistic Non-Independent and Identically Distributed (non-IID) data. This chapter presents **C5**.

Chapter7 - Conclusions and Future Work: This chapter concludes by reflecting on the contributions made, highlighting the main challenges, and exploring potential future directions.

Chapter 2

Background: CAN: Functionality, Vulnerabilities and Attacks

An overview of the in-vehicle network and the CAN protocol in the context of this research, along with its vulnerabilities, entry points, and attack scenarios, is presented. The objective is to introduce key terminology and provide an understanding of the CAN bus protocol while exploring potential attack scenarios and their impacts.

2.1 Introduction

This chapter provides a brief overview of the in-vehicle network and its protocols, with a particular focus on the CAN protocol, including its description, functionality, and aspects relevant to cyberattacks. It further examines CAN vulnerabilities, entry points, and attack scenarios. The objective of this chapter is to understand the CAN bus protocol within the context of this research and analyse how different attacks are executed by exploring various attack scenarios.

2.2 In-vehicle Network

The in-vehicle network is considered the backbone of CAVs in modern automotive systems [46]. It serves as an internal communication system that connects multiple embedded devices within the vehicle [47]. These devices, known as Electronic Control Units (ECUs), are responsible for managing various vehicle functions such as engine control, airbag deployment, and climate regulation. The number and type of ECUs vary depending on the manufacturer

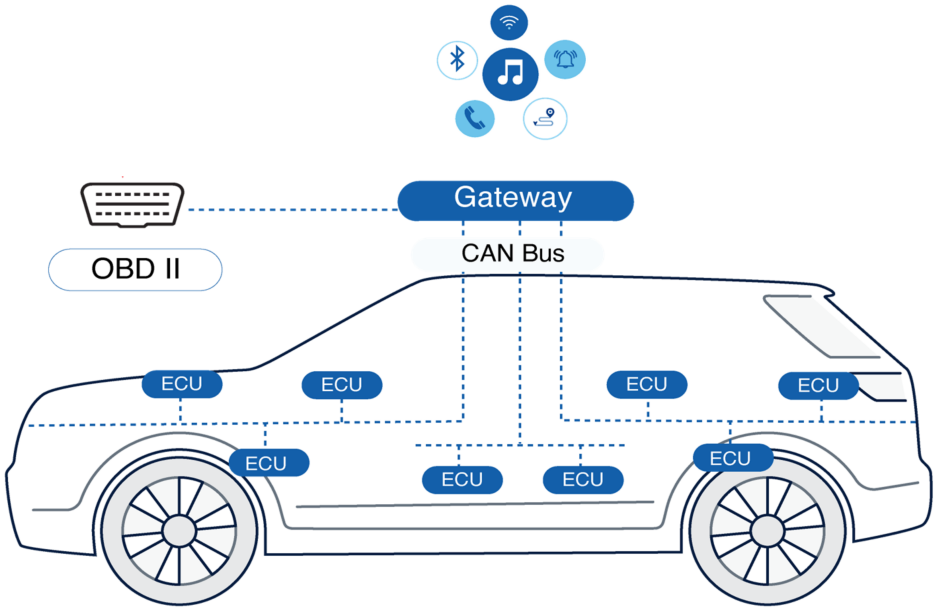


Fig. 2.1 Overview of in-vehicle network.

and model, with modern vehicles incorporating up to 100 ECUs alongside basic functionalities [48]. Although ECUs are essential for vehicle operation, they also pose significant security risks, as attackers can exploit them to sniff traffic or inject malicious data into the bus [12]. Figure 2.1 shows an overview of the in-vehicle network.

These ECUs, along with sensors, actuators, cameras, radars, and communication devices, collaborate to enhance vehicle performance, efficiency, intelligent services, and safety by collecting and interpreting various data [49]. ECUs communicate using standard protocols such as CAN, FlexRay, Local Interconnect Network (LIN), and Media Oriented System Transport (MOST) [50]. Among these, CAN is considered the de facto protocol for in-vehicle communication [7].

2.3 Controller Area Network

CAN is a serial bus communication protocol designed for in-vehicle applications, developed by Robert Bosch in 1985 and standardised in the ISO 11898 series [51]. It was designed to reduce the weight, complexity, and cost of wiring. Due to its high speed and efficiency, CAN has become the most widely used in-vehicle communication protocol in CAVs [7]. CAN operates as a message-based broadcast protocol, where the ECUs transmit data in pre-defined

frames. Since the system uses a broadcast mechanism, each message is sent to all ECUs on the network.

2.4 CAN Bus Data Frame

The CAN frame follows a specific message structure defined in a database-like file known as the DataBase CAN (DBC) file. This file is confidential and proprietary to the vehicle manufacturer, containing all the essential information about the representation of the CAN bus data [21]. The CAN data frame consists of seven fields that facilitate data transmission between ECUs. Figure 2.2 illustrates the standard CAN frame format, which consists of the following fields:

- **Start of Frame (SOF):** The purpose of this field is to indicate the start of CAN frame transmission to other nodes using a dominant '0' bit.
- **Arbitration Field:** This field consists of the Identifier (ID), which is used to specify the destination address of the designated ECU. It also determines the priority of the message, where a lower value generally indicates higher priority. The ID field is 11 bits in the standard format and 29 bits in the extended format. The field also includes the Remote Transmission Request (RTR) bit, which distinguishes between data frames and remote frames.
- **Control Field:** This field is 6 bits long and includes a 4-bit Data Length Code (DLC), which indicates the length of the data field, an Identifier Extension (IDE) bit, which specifies whether the ID field is standard (11 bits) or extended (29 bits), and a Reserved Bit (RB) for future use.
- **Data Field:** This field, also known as the payload, includes the actual vehicle parameter values, which are interpreted by the received ECU and its size can vary from 0 to 8 bytes (0-64 bits).
- **Check Field:** This field consists of a 15-bit Cyclic Redundancy Check (CRC) followed by a 1-bit Delimiter (DEL), and is used to detect errors and maintain data integrity during message transmission by verifying the validity of the frame.
- **Acknowledge Field (ACK):** This field consists of 2 bits: a 1-bit ACK and a 1-bit DEL. The ACK bit is used to receive confirmation from the receiving node that the CAN message was received correctly.

- **End of Frame (EOF):** This field indicates the completion of CAN message transmission.

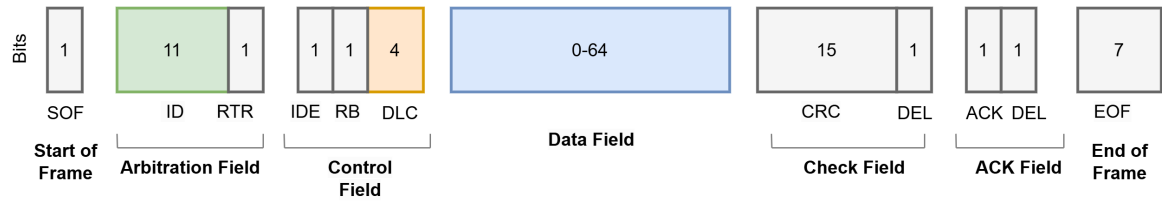


Fig. 2.2 CAN data frame.

In this thesis, we focus on the coloured fields (ID, DLC, and Data) shown in Figure 2.2. Among the various CAN frame fields, the CAN ID and the data field are the most commonly targeted for exploitation. Although the DLC can technically range from 1 to 8, values outside this range are invalid and cannot be used or modified [4].

2.5 CAN Vulnerabilities

The CAN bus was introduced to reduce costs, simplify installation, and improve real-time communication efficiency within vehicles. However, it is vulnerable to cyberattacks due to several inherent vulnerabilities [12, 52, 53], including the following:

- *Lack of authentication:* Due to the lack of authentication on the CAN bus, any ECU can transmit a frame using the CAN ID of another ECU [14]. Each ECU broadcasts and receives all data on the bus, then determines whether a message is intended for it. However, the CAN protocol is inherently unable to prevent unauthorised devices from joining the network and sending malicious messages to all ECUs. As a result, attackers can exploit compromised ECUs to spoof and send fake CAN packets, leading to spoofing and message injection attacks [27].

- *Lack of encryption:* Due to time constraints, CAN messages are not encrypted [12], allowing cyberattackers to easily capture and analyse them. The lack of encryption makes CAN traffic vulnerable to sniffing, spoofing, modification, and replay attacks [14].

- *Broadcast domain:* The CAN bus functions as a broadcast domain, where all ECUs receive the transmitted frames. Each ECU then checks the data and determines whether to process or disregard it [48]. If an ECU is compromised, it can intercept and monitor

all messages transmitted across the CAN network, enabling an eavesdropping attack [54].

- *ID-based priority:* The CAN network prioritises messages based on their IDs, with lower IDs having higher priority [27]. Attackers can exploit this by repeatedly sending frames with low IDs, resulting in a Denial-of-Service (DoS) attack [21].
- *Unsegmented Network:* All ECUs are connected to a single shared network without segmentation [14], which is a key reason why CAN was adopted in automotive systems, as it eliminates the need for point-to-point connections. However, this shared network allows components such as infotainment systems to communicate with safety-critical vehicle systems. Although some manufacturers use separate networks for safety-critical systems, there is still cross-communication between critical and non-critical systems [14].
- *External Interfaces:* The attack surface of the CAN bus network is expanded by external interfaces such as the Onboard Diagnostic(OBD)-II Port, used for vehicle maintenance and diagnostics; the Telematics Unit, which provides connectivity to the vehicle via Wi-Fi, Bluetooth, GPS, and mobile data interfaces; and the Infotainment Unit, which delivers information and entertainment to the driver through a head display unit, including features like CD/DVD players and USB ports. These interfaces create additional entry points for potential cyberattacks [55, 27].

2.6 In-vehicle Network Entry Points

Attackers can gain access to the CAN bus or specific ECUs either physically or remotely [12, 46, 56, 52]. These entry points serve as gateways for initiating a range of attacks, exploiting the inherent vulnerabilities described in Section 2.5 in in-vehicle networks. This section discusses the entry points attackers can exploit to access the in-vehicle network, either physically or remotely.

2.6.1 Physical Access

Physical access allows an attacker—such as a mechanic, valet, car renter, or anyone with even brief access to the vehicle—to directly interact with its internal systems. This access, even for a short time, can provide opportunities to exploit vulnerabilities through various physical entry points, such as the OBD-II port and aftermarket components.

- **Onboard diagnostic(OBD)-II Port:** The OBD-II port, commonly located under the dashboard in most vehicles, provides the simplest and most direct access to a vehicle's primary CAN buses. This port offers sufficient access to potentially compromise the full range of automotive systems [56]. Designed primarily for vehicle maintenance and engine diagnostics, the OBD-II port allows mechanics to connect scanning tools and capture data packets generated by malfunctioning subsystems. Despite its intended purpose, the port's accessibility makes it a significant security vulnerability. Attackers can easily connect to the OBD-II port to extract information or install malware onto the vehicle's systems, disconnecting afterward to leave no physical evidence [57]. Alternatively, attackers may deploy a remote device to the port, or enabling continuous data collection or exploitation over time. Since the OBD-II port is required for maintenance and diagnostics, it will always pose a security risk [52].

- **Aftermarket Components:** Peripheral components such as USB ports, CD players, and third-party add-ons also pose security risks [53, 52]. For example, malicious devices, including FM radios, USB connectors, or CD players purchased from unverified or aftermarket sources, can introduce malware into the vehicle's system. While these components may be more affordable, they can compromise the vehicle's security [57].

2.6.2 Remote Access

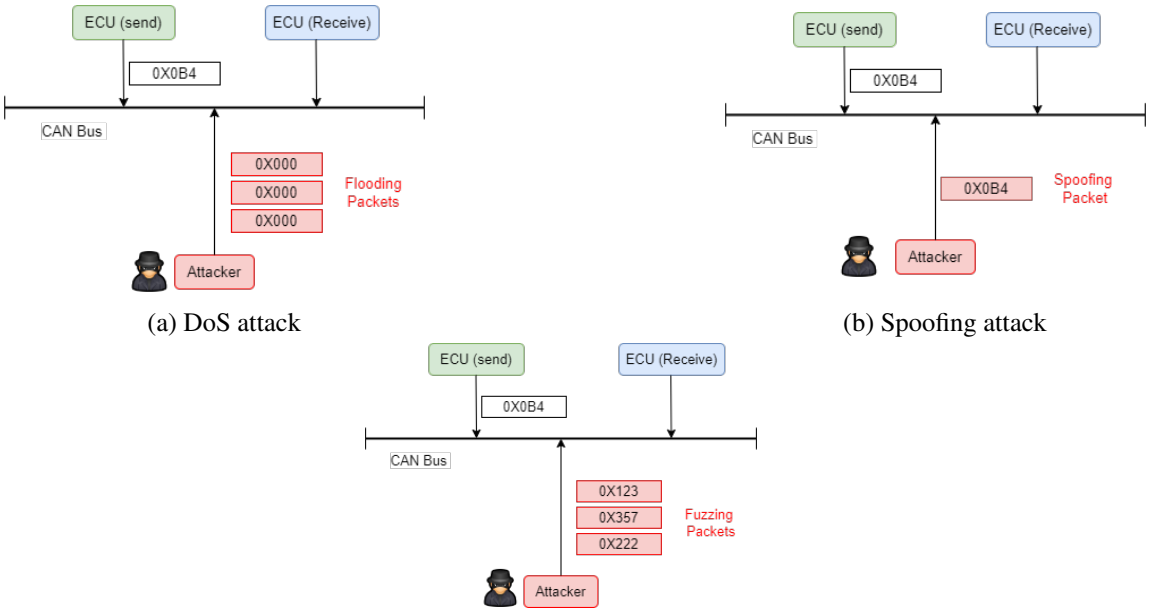
An external attacker can exploit wireless interfaces commonly implemented in modern vehicles, such as Bluetooth, Wi-Fi, cellular networks, and GPS, without requiring physical proximity to the vehicle. Once these interfaces are accessed, the attacker can inject malicious data or commands into the CAN bus [12]. Koscher et al. [26] highlight the feasibility of executing various types of wireless attack injections on in-vehicle network systems. For example, vulnerabilities in telematics systems or vehicle-to-cloud communications can enable the remote injection of messages, disrupting the network. Specific methods include using malicious Windows Media Audio (WMA) files or sending malicious packets to the telematics unit via 3G Internet Relay Chat (IRC) [26]. Moreover, Woo et al. [58] conducted a wireless attack, successfully taking control of a target vehicle by utilising malware installed on a smartphone. These examples highlight the significant security risks posed by wireless interfaces in connected vehicles.

2.7 Attack Scenarios

In this section, we present the attack scenarios considered in this thesis. Specifically, we focus on three types of manipulation attacks targeting the CAN ID and/or data field: DoS, spoofing, and frame fuzzification. These attacks were selected because they involve manipulation of CAN identifiers, payloads, or both, thereby covering the full spectrum of CAN frame content manipulation. Prior studies have demonstrated that such attacks are realistic and pose serious risks to the safety of drivers, passengers, and other road users [59–61]. In addition, Abuabed et al. [62] ranked these attacks as high risk within a STRIDE-based threat modelling framework, where DoS attacks were prioritised due to their severe impact on the availability of safety-critical CAN traffic and their relatively high feasibility. Fuzzing was ranked above spoofing because it combines moderate to high feasibility with a broader potential to corrupt multiple message types, whereas spoofing typically targets specific ECUs or functions. In contrast, attacks that exploit message timing rather than content, such as replay attacks, were excluded from this study because their detection requires temporal analysis, which lies beyond the scope of this work. Moreover, CAN messages may follow either periodic or event-driven transmission patterns, meaning that reliable detection of timing-based attacks depends on precise timing specifications. These specifications are defined in the DBC file, which is proprietary to vehicle manufacturers [27]. Throughout this section, we assume that the attacker has gained access to the in-vehicle network, either physically or remotely, through one of the entry points outlined in Section 2.6.

2.7.1 Denial-of-Service (DoS) Attack

- **Attack Definition:** The goal of a DoS attack is to overwhelm the CAN bus bandwidth by transmitting large volumes of messages, leading to system malfunctions and service disruptions [27]. Koscher et al. [57] demonstrated that DoS attacks can disable individual CAN bus components.
- **Attack Method:** Since message priority is determined by the arbitration field, an attacker can exploit the CAN frame priority arbitration scheme vulnerability by sending numerous messages with low CAN IDs (high priority), such as 0x0000. This flooding of high-priority frames occupies the bus, preventing other ECUs from accessing it [63]. Figure 2.3a illustrates how a high-priority CAN ID 0x0000 delays a lower-priority CAN ID 0x0B4.



(a) DoS attack

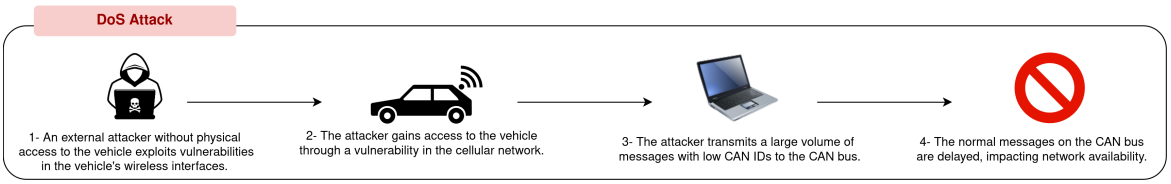
(b) Spoofing attack

(c) Frame fuzzification attack

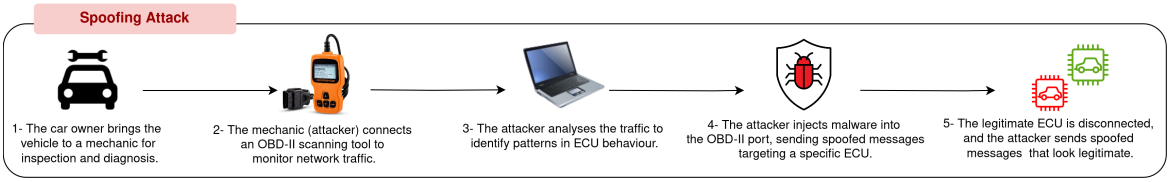
Fig. 2.3 CAN bus attacks

- **Attack Scenario:** We assume that the attacker has gained access to the in-vehicle network, either physically or remotely, through one of the entry points outlined in Section 2.6. Leveraging this access, the attacker floods the CAN bus with high-priority messages, such as those with CAN IDs like 0x0000, without requiring prior knowledge of the CAN bus traffic. The arbitration mechanism prioritises these malicious messages, taking control of the bus and blocking critical communications, such as those from the engine control unit or braking system. For instance, while the vehicle is in motion, an attacker carrying out this attack could disable cruise control or activate emergency braking, preventing critical messages from reaching the appropriate ECU in time and creating potentially hazardous driving conditions. Within seconds, the network's capacity becomes overwhelmed, causing delays that severely compromise vehicle safety. Figure 2.4a illustrates an example of a DoS attack scenario carried out by an attacker.

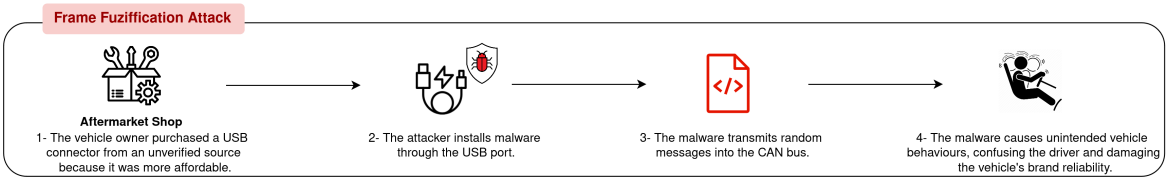
- **Attack Impact:** A successful DoS attack not only delays normal messages by occupying the bus [61], but also prevents other ECUs from transmitting frames to the in-vehicle network, significantly impacting network availability [64]. Such attacks can lead to a complete breakdown of ECU communication and severe disruption of the entire CAN bus network system [65, 53], posing significant threats to the safety of drivers, passengers, and other road users [59].



(a) DoS attack scenario.



(b) Spoofing attack scenario.



(c) Frame fuzzification attack scenario.

Fig. 2.4 Attack scenarios.

2.7.2 Spoofing Attack

- **Attack Definition:** In a spoofing attack, an unauthorised attacker targets specific existing CAN IDs and injects fabricated messages to control particular functions. Since CAN IDs appear legitimate, distinguishing between real and spoofed messages becomes challenging, leading to system malfunctions [63].

- **Attack Method:** The attacker may sniff the CAN bus traffic or possess prior knowledge about the ECUs' CAN messages. They can then use this information to inject spoofed messages into the CAN bus. Figure 2.3b illustrates a spoofing attack where an attacker, using the spoofed CAN ID 0x0B4, targets the legitimate CAN ID 0x0B4. This enables the attacker to disrupt vehicle functions by generating manipulated messages that appear legitimate.

- **Attack Scenario:** We assume that the attacker has gained access to the in-vehicle network, either physically or remotely, through one of the entry points outlined in Section 2.6. In this attack, we assume that the attacker has some knowledge of CAN bus traffic by implementing an impersonation attack. One method to achieve this is by connecting a malicious device to eavesdrop on all broadcast traffic, capturing data transmitted across the network. During this reconnaissance phase, the attacker

analyses the traffic to identify patterns in ECU behaviour, such as specific CAN IDs, payload structures, and message transmission intervals. Armed with this knowledge, the attacker selects a target ECU, such as the speedometer, with plans to disable it from the bus later. Next, the attacker gains remote access to the internal network and crafts spoofed messages by replicating the target ECU's CAN ID and injecting false speed readings into the bus. This activity exploits the CAN bus protocol's error-handling mechanisms, as described in [60]. By transmitting dominant bits whenever the legitimate ECU sends recessive bits, the attacker creates intentional bit conflicts. These conflicts generate repeated error frames, eventually causing the legitimate ECU to exceed its error tolerance threshold and enter a "bus-off" state. In the bus-off state, the legitimate ECU is effectively disconnected from the network, unable to send or receive critical messages. The attacker then injects maliciously crafted messages using the same CAN ID as the target ECU. For instance, they could dangerously slow the vehicle on a highway or increase its speed in restricted areas, thereby creating hazardous conditions. Figure 2.4b illustrates an example of a spoofing attack scenario carried out by an attacker.

- **Attack Impact:** Spoofing attacks can cause system malfunctions and disrupt vehicle operations [63]. They pose significant threats to personal safety, particularly when targeting critical ECUs responsible for essential functions such as braking or steering [60].

2.7.3 Frame Fuzzification Attack

- **Attack Definition:** The goal of a frame fuzzification attack is to inject random messages into the CAN bus network, making them appear as legitimate traffic. Attackers may exploit prior knowledge of CAN IDs and payload values obtained through CAN bus sniffing, or they may perform the attack without any prior knowledge of CAN frames, treating it as a black-box attack [66]. In this type of attack, the attacker might alter the CAN ID, the CAN payload, or both simultaneously [12]. Since the range of valid CAN packets is relatively small, even simple fuzzing of packets can cause significant damage [57].

- **Attack Method:** The attacker sends arbitrary messages into the CAN bus network, making them appear as legitimate traffic. Figure 2.3c illustrates a frame fuzzification attack, where the attacker generates and injects random CAN IDs (e.g., 0x123, 0x357,

Table 2.1 CAN bus attacks.

Attack	Manipulated Feature	Violated Security Property	Vulnerability	Consequences
DoS	CAN ID	Availability	No authentication, Message ID priority scheme	Network failures [55], prevent legitimate communication between ECUs from using the bus [12]
Spoofing	CAN Payload	Integrity	No authentication, No encryption	Engine control, brake control, steering wheel control [67], airbag detonation [68]
Frame fuzzification	CAN Frame	Integrity	No authentication	Wheel shaking, unpredictable signal light switching, automatic gear shifts [61], disrupt vehicle operations [69]

and 0x222), which are illegitimate. As a result, all ECUs receive a high volume of functional messages, leading to unintended vehicle behaviours [61]. For example, Chockalingam et al. [26] introduced Gaussian noise to create a frame fuzzification attack on CAN data. A frame fuzzification attack can disrupt the entire CAN bus network, leading to severe malfunctions such as the steering wheel shaking uncontrollably, signal lights flickering erratically, or automatic and unintended changes in the gear shift [65].

- **Attack Scenario:** We assume that the attacker has gained access to the in-vehicle network, either physically or remotely, through one of the entry points outlined in Section 2.6. Without prior knowledge of CAN frames, the attacker is able to inject random malicious CAN frames. Using techniques such as fuzzing, the attacker transmits random or malformed messages into the CAN bus to provoke unintended system behaviours or identify exploitable vulnerabilities that could be leveraged in future attacks. Additionally, through reverse engineering, the attacker monitors legitimate traffic to deduce the structure and purpose of CAN messages, enabling the creation of malicious packets to execute specific commands targeting particular ECUs. Figure 2.4c illustrates an example of a frame fuzzification attack scenario carried out by an attacker.

- **Attack Impact:** Frame fuzzification attacks can compromise ECUs, triggering unexpected vehicle behaviours such as steering wheel shaking, erratic signal lights, and unintended gear shifts [61]. These behaviours can confuse the driver, potentially resulting in poor decisions or accidents. Such attacks not only disrupt normal vehicle functions but also threaten operational integrity, compromise data privacy, and endanger personal safety, posing significant risks to passengers and other road users.

Table 2.1 summarises the previous sections by detailing each CAN bus attack, the targeted feature, the violated security property, the underlying vulnerability, and the associated consequences.

2.8 Conclusion

In this chapter, we provided an overview of the in-vehicle network and the CAN bus protocol, emphasising its crucial role in in-vehicle communication. However, the CAN bus is inherently vulnerable due to the lack of essential security mechanisms, such as sender authentication and encryption, which make it susceptible to a range of cyberattacks. These vulnerabilities not only jeopardise information security and privacy but also pose significant risks to the physical safety of drivers, passengers, and others. We also explored various attack scenarios that can be executed on the CAN bus, their entry points, and their impact. Therefore, enhancing the security of in-vehicle networks is an urgent priority. In the next chapter, we review the efforts made to detect cyberattacks in in-vehicle networks.

Chapter 3

Literature Review [1]

This chapter presents a comprehensive literature review of current in-vehicle ML and DL-based IDSs for detecting known, unknown, and combined known-unknown attacks, as well as FL-based IDSs. We employed a structured search strategy to systematically gather research papers published up to January 2025. The aim is to identify limitations and research gaps in the existing work. Based on these gaps, open research questions are formulated, and the scope of the thesis is outlined.

3.1 Introduction

In this chapter, we review the current research on ML, DL, and FL based in-vehicle IDSs, aiming to identify limitations and research gaps in the existing work. The chapter is organised as follows: First, we provide a brief overview of in-vehicle IDSs, their various approaches, and the FL approach. Then we present the search strategy employed to gather research papers. Based on the collected papers, we examine current in-vehicle IDS approaches, categorising previous work based on the types of attacks they detect: known, unknown, and combined known-unknown attacks. We also identify the limitations of each approach and key research questions. Next, to further enhance the robustness of in-vehicle IDSs, we review FL-based IDSs, highlighting their limitations and emerging research questions.

[1] This chapter is published in *Computer Networks* (2026): 112031.

3.2 Background

In this section, we begin by introducing IDSs for in-vehicle networks and highlighting the differences between in-vehicle IDSs and those used for other applications. We then provide an overview of in-vehicle IDS approaches and outline the focus of this thesis. Lastly, we provide an overview of the FL approach within the context of in-vehicle networks.

3.2.1 Overview of IDSs for In-Vehicle Networks

According to NIST SP 800-94, intrusion detection is "the process of monitoring the events occurring in a computer system or network and analysing them for signs of possible incidents" [70]. Therefore, an IDS is typically considered a software or hardware system designed to automatically detect suspicious activity in a network [14]. In vehicle networks, IDSs are crucial for identifying malicious attacks [12]. They can be implemented as either host-based or network-based systems [14]. Host-based IDSs are installed on each vehicle's ECU, allowing comprehensive monitoring of internal ECU operations. In contrast, network-based IDSs are deployed within the CAN network or central gateways to oversee all network traffic. However, Host-based IDSs are not a viable solution for vehicles, as they require a change in ECUs that are not cost-effective [12]. In contrast, deploying a network-based IDS as an additional node on the CAN bus is a more feasible and practical solution, as it avoids the need for any CAN bus modifications [27]. Unlike IDSs in other applications, in-vehicle IDSs are constrained by computing power, memory size, and communication capabilities. This is because modern ECUs in vehicles are primarily powered by 32-bit embedded processors, with limited computational performance and memory resources [55].

3.2.2 In-Vehicle IDS Approaches

Research on developing in-vehicle IDSs has grown significantly in recent years, driven by the critical need to enhance the security of in-vehicle networks and detect cyberattacks. Researchers have explored various approaches to building these systems. IDSs can be classified as either signature-based, for detecting known attacks, or anomaly-based, for identifying new, unknown attacks [71]. Anomaly-based IDSs are further categorised into statistical, ML, rule-based, and physical fingerprinting methods [27]. However, this work specifically focuses on the development of in-vehicle IDSs using ML and DL approaches. This focus arises from the widespread use of ML and DL-based IDSs to process large volumes of CAN traffic data. These approaches efficiently extract and pre-process raw CAN

data, which is critical as vehicle manufacturers often do not provide detailed specifications for decoding these raw data [27].

3.2.3 Overview of Federated Learning

FL is a privacy-preserving decentralised learning technique that trains models locally without transferring the row data to a centralised server [72]. Instead, it transfers model parameters to a centralised server, which aggregates the clients' models to build a shared global model [73]. This integration of FL into IDSs enhances security and privacy, addressing the growing challenges of protecting data in an increasingly interconnected world. While ML and DL have made notable progress in in-vehicle IDSs, it is crucial to recognise their limitations, particularly regarding data privacy and communication efficiency. FL mitigates these challenges by enabling local model training while preserving the privacy of raw data [34]. FL is increasingly adopted across various industries, such as healthcare [74], finance [75], and cybersecurity [76], due to its privacy-preserving capabilities. FL is well-suited for in-vehicle IDSs for several compelling reasons:

- The FL approach preserves data privacy by periodically transmitting learned model parameters to the cloud server instead of sharing raw data [70]. This aligns with various data protection regulations, such as the General Data Protection Regulation (GDPR, Europe) [77], the California Consumer Privacy Act (CCPA, California) [78], the Personal Information Protection and Electronic Documents Act (PIPEDA, Canada) [79], and the Lei Geral de Proteção de Dados (LGPD, Brazil) [80], which are designed to prevent the unauthorised transfer of sensitive information.

- FL provides fast analysis, which is crucial for supporting highly dynamic, time-critical applications, whereas centralised processing is time-consuming [73].

- FL reduces latency by training data locally and transmitting only model updates, FL significantly reduces latency compared to traditional centralised approaches [76, 73], which require sending raw data to a central server [70].

- Referring to the 2020 guidelines of the International Telecommunication Union [81] for IDS in vehicular networks, an in-vehicle IDS must have the ability to regularly update its set of rules. FL supports this requirement by enabling continuous learning and iterative model refinement.

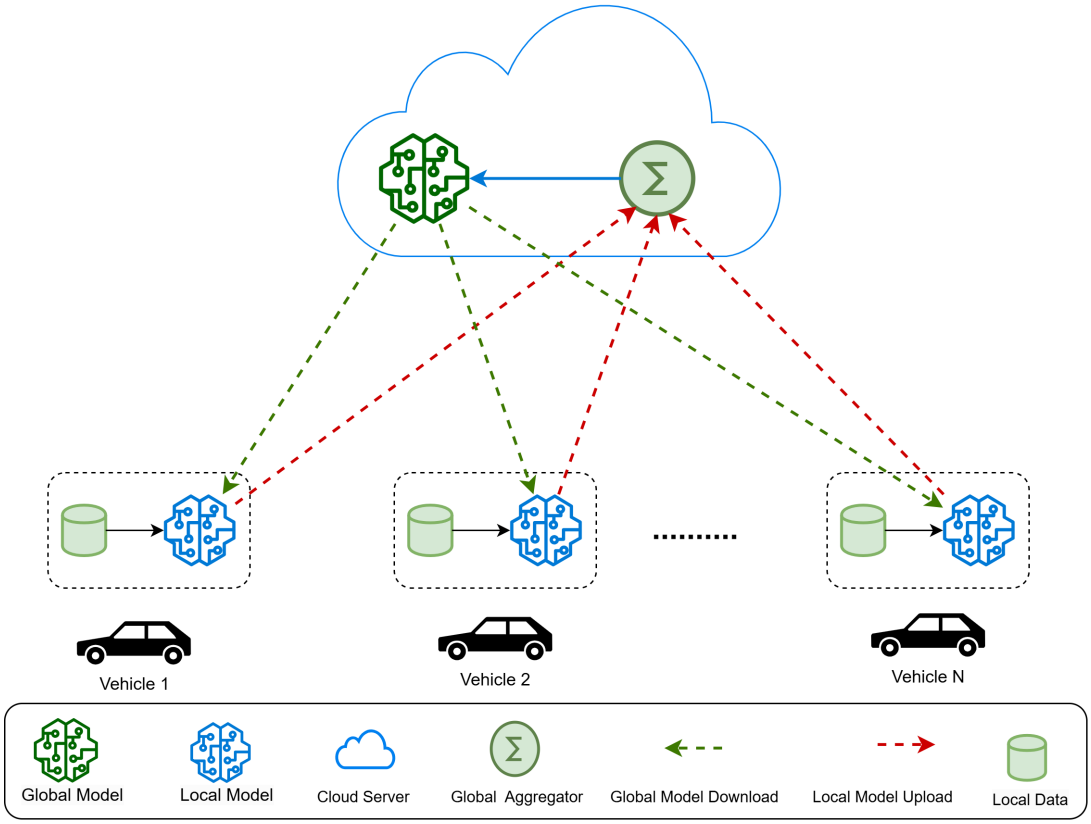


Fig. 3.1 Federated learning architecture.

- FL allows multiple participants to collaboratively develop a robust global model while preserving user data privacy, enabling the creation of a universal model that captures diverse driving scenarios, vehicle states, and driving behaviours [38].

As depicted in Figure 3.1, the standard cloud-based FL architecture consists of a cloud server and multiple N clients (vehicles). Selected clients download the global model from the server, perform several rounds of local training using their own private data, and subsequently return the updated model weights to the server for aggregation. This iterative process continues until the model reaches the desired level of accuracy.

3.3 Search Strategy

In this section, we outline the search strategy used to conduct our literature review. To select the studies for inclusion, we followed Kitchenham's [82] method, a well-established and effective guide for identifying relevant literature. Although originally designed for the software engineering domain, this method has been widely applied in other fields, including

cybersecurity [83]. Our process began with an automatic search using Google Scholar to minimise bias towards any specific publisher [84] and to identify key publishers and conferences in the field. Based on this search, we compiled a list of publishers and conferences for a subsequent manual search. To ensure comprehensive coverage, we employed a snowballing approach to locate all related papers. After gathering a substantial number of publications, we applied filtering processes to select the most relevant ones. Finally, we analysed the collected papers (from any time up to and including January 2025) to develop the literature review presented in this chapter. Figure 3.2 depicts the adopted search strategy.

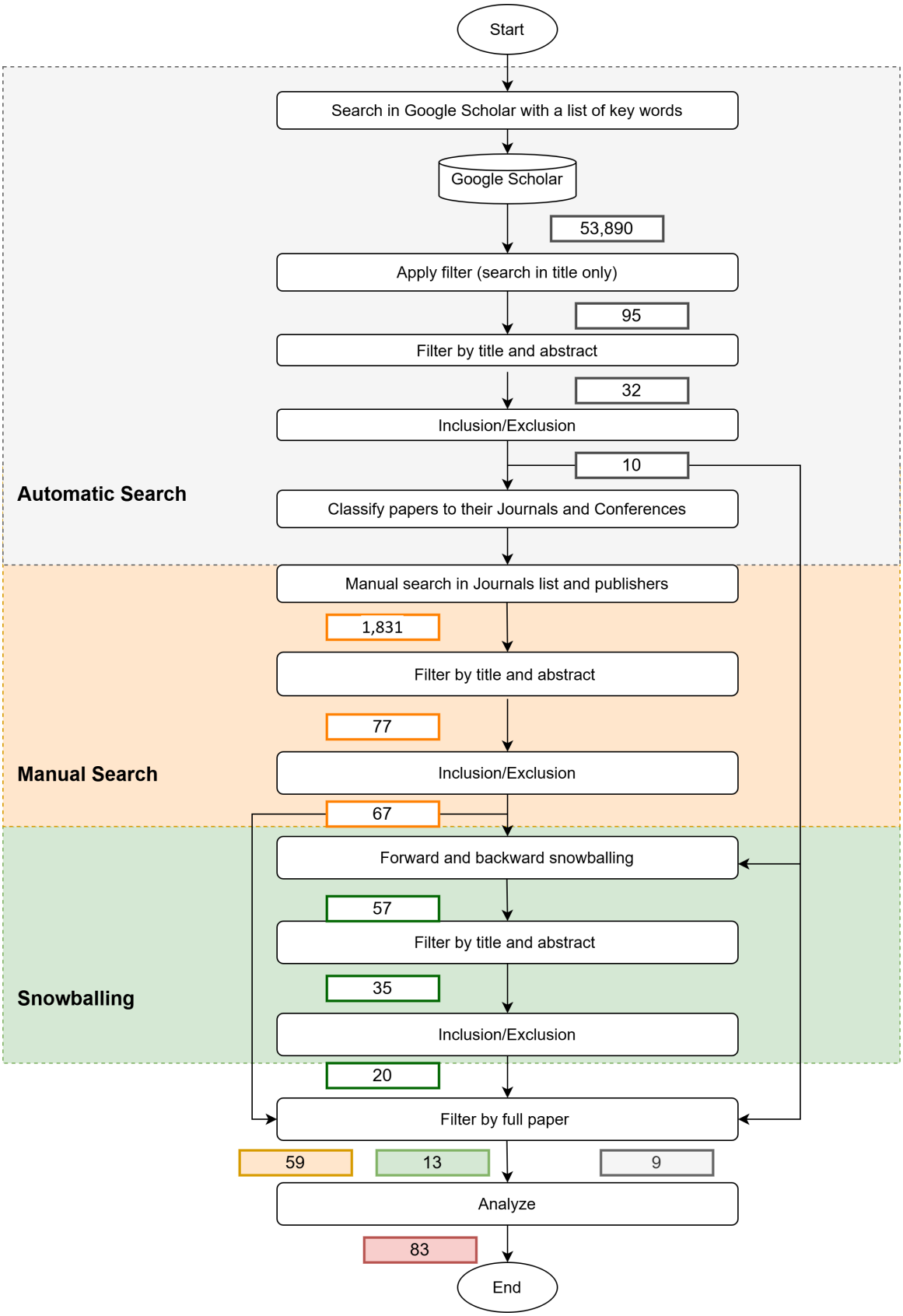


Fig. 3.2 Search and selection processes flowchart.

3.3.1 Data Sources and Search Strategy

To begin, we formulated several search queries on Google Scholar using keywords, drawn from papers across all years, that combine domain-specific terms with those commonly used in article keywords. These terms are listed in Table 3.1.

Table 3.1 Google Scholar search terms and results.

Key Terms	Anywhere in the Article	In the Title
("CAN bus" OR "Controller Area Network") AND ("IDS" OR "intrusion detection system")	9,220	10
"In-vehicle" AND ("IDS" OR "Intrusion Detection System")	20,000	25
("CAN bus" OR "Controller Area Network") AND "anomaly detection"	4,690	11
"In-vehicle" AND "anomaly detection"	9,970	39
("CAN bus" OR "Controller Area Network") AND "unknown attacks"	658	0
"In-vehicle" AND "unknown attacks"	892	0
("CAN bus" OR "Controller Area Network") AND "federated"	1,860	0
"In-vehicle" AND "federated"	6,600	10
Total	53,890	95

Logical operators "AND" and "OR" were utilised to ensure comprehensive results. These queries generated a range of papers, some of which were only loosely relevant. This step also provided insights into the digital libraries and journals that prioritise CAN bus security. Subsequently, we conducted a hybrid search using more complex queries in specific journals and libraries, including IEEE Xplore, Scopus, ACM, MDPI, Springer, and ScienceDirect. Three search strategies were employed during this process: automatic, manual, and snowballing.

1. **Automatic search** We carried out this stage using the advanced search function in Google Scholar, employing key terms such as "CAN bus", "controller area network",

"in-vehicle", "intrusion detection system", "IDS", "anomaly detection", "unknown attacks", and "federated". These root words were chosen because Google Scholar automatically searches for variations of the same word; for instance, searching for "federated" returns results like "federated", "federated learning", "federated-based", and "federated environment". Additionally, the search was not restricted to a specific time period. Initially, using the filter "anywhere in the article," we retrieved an unwieldy number of results (53,890) (see Table 3.1). To refine this, we applied the filter "in the title of the article," reducing the results to 95 papers. Only peer-reviewed papers were included in our analysis.

2. **Manual search** In this stage, we applied more complex queries, including specific journals and libraries, such as IEEE Xplore, Scopus, ACM, MDPI, Springer, and ScienceDirect. Table 3.2 lists examples of queries used in the manual search.

3. **Snowballing** The snowballing technique was applied to the papers identified through automatic and manual searches. This approach included both forward and backward snowballing. Forward snowballing (or citation analysis) locates papers that are cited in the papers found in the initial stages. Backward snowballing (or reference analysis) looks at the reference lists of the papers found in the initial search process. References included in the selected papers were chosen based on a review of the title, abstract, and the paper's structure. We found that backward snowballing using critical papers was an effective means of identifying relevant papers. The set of articles selected was updated to include any additional relevant studies found by snowballing.

3.3.2 Selection Strategy

The selection strategy involved defining inclusion and exclusion criteria, as well as applying filtering during the search process. The steps in the selection process included applying the inclusion and exclusion criteria, followed by an additional filtering stage involving a quality assessment to ensure the selection of high-quality studies. Each of these steps is discussed below.

- **Filtering Irrelevant Papers**: The papers collected through manual, automatic, and snowballing approaches included several that did not apply directly to our study and had to be eliminated. Elimination was conducted in two steps. First, papers were excluded based on the title, keywords, abstract, and, in cases of doubt, the conclusion.

Table 3.2 Examples of queries and terms used for the online library search.

Category	Queries and Terms
General	"CAN bus" OR "controller area network" OR "in-vehicle" AND "intrusion detection system" OR "IDS" OR "anomaly detection" OR "unknown attacks" OR "federated".
More Specific	**ACM:** [[Title: "can bus"] OR [Title: "controller area network"] OR [Title: "in-vehicle"]] AND [[Title: "intrusion detection system"] OR [Title: "ids"] OR [Title: "anomaly detection"] OR [Title: "unknown attacks"] OR [Title: "federated"]] **IEEE Xplore:** ("Document Title":"CAN bus") OR ("Document Title":"controller area network") OR ("Document Title":"in-vehicle") AND ("Document Title":"intrusion detection system") OR ("Document Title":"IDS") OR ("Document Title":"anomaly detection") OR ("Document Title":"unknown attacks") OR ("Document Title":"federated") **Scopus:** (TITLE-ABS-KEY ("CAN bus" OR "controller area network" OR "in-vehicle") AND TITLE-ABS-KEY ("intrusion detection system" OR "IDS" OR "anomaly detection" OR "unknown attacks" OR "federated"))

Based on this step, a decision was made regarding whether to include each paper in the next step. Eliminating was undertaken after each search (automatic, manual, and snowballing) to reduce the number of papers. When a paper passed the initial elimination step, it was subjected to the inclusion and exclusion criteria.

- **Inclusion and Exclusion Criteria** In this process, we defined our exclusion and inclusion criteria. A paper was considered for exclusion if it met one or more criteria. The exclusion criteria for each paper were as follows: (i) not written in English; (ii) was a review or survey paper; (iii) lacked a full version (e.g., only a poster or abstract); (iv) did not employ an ML or DL approach; (v) required reverse engineering; (vi) used other data alongside CAN bus data; and (vii) employed other approaches such as statistical, rule-based, or physical fingerprinting methods. Papers that were not excluded were then evaluated according to an inclusion list of other criteria. If no inclusion criteria were met, the paper was rejected. The inclusion criteria were as follows: (i) focused on the CAN bus protocol rather than other in-vehicle protocols; and (ii) focused on attack detection as the primary goal. Non-peer-reviewed papers were included only if they were strictly relevant to the topic and had a high citation rate, or if the author was well-known in the field.

As shown in Figure 3.2, reading the full paper is the final step in the search and selection process, filtering out the collected papers from the previous steps. The papers from the automatic search were reduced from 10 to 9 after filtering by reading the full text. In the manual search within journal lists and publishers, the initial 1,831 papers were filtered down to 59. For snowballing, 57 papers were initially selected and reduced through filters to 15. Thus, the total number of collected papers comprises 9 from the automatic search, 59 from the manual search, and 15 from snowballing, resulting in a total of 83 papers for analysis. Of these, 38 focused on known attack detection, 27 on unknown attack detection, 10 on IDSs capable of detecting both known and unknown attacks, and 8 on FL-based IDSs. Figure 3.3 illustrates the number of collected papers in each category, highlighting that known attack detection is the most researched area, while significantly less work has been conducted on IDSs capable of detecting both known and unknown attacks, as well as on FL-based IDSs.

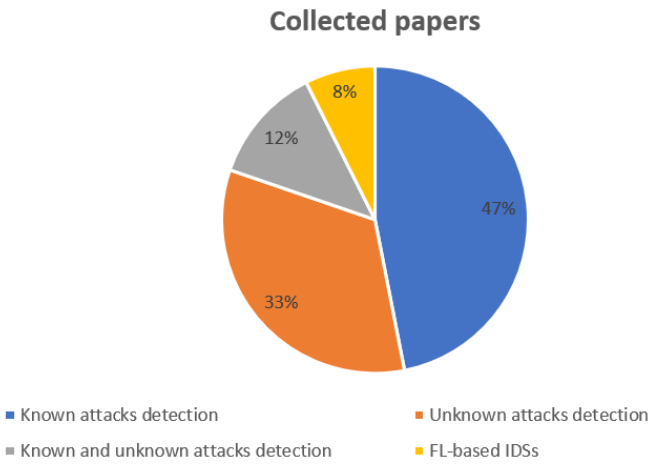


Fig. 3.3 Distribution of collected papers by category.

3.4 Related Work on ML/DL-Based In-Vehicle IDSs

In this section, we categorise the collected papers into three groups for analysis: known attack detection, unknown attack detection, and approaches capable of detecting both. Figure 3.4 illustrates the categories and subcategories of the reviewed literature. The limitations of each approach are highlighted. In addition, we review all the evaluation metrics used in the reviewed papers.

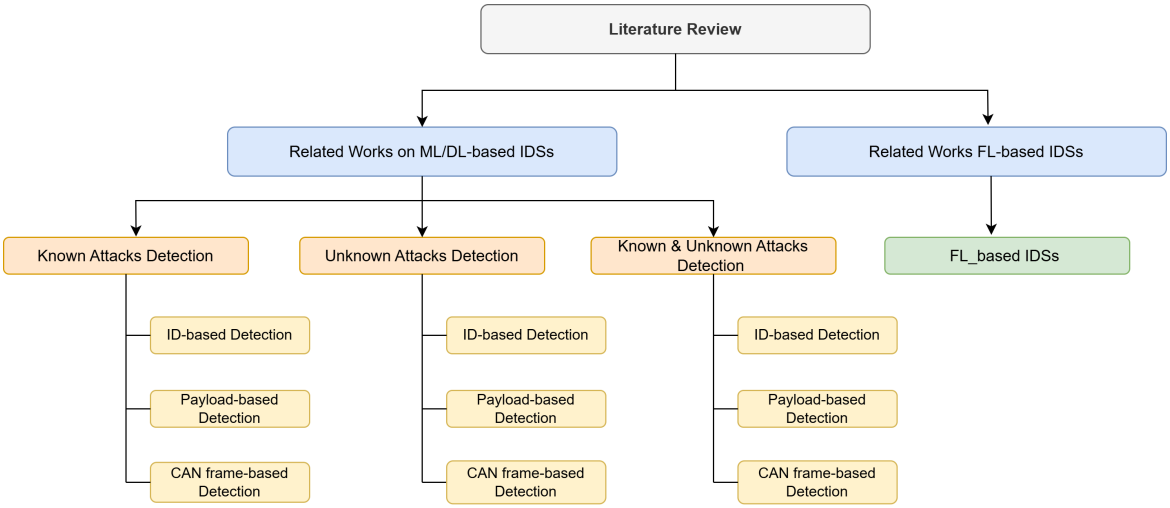


Fig. 3.4 Categories of reviewed literature.

3.4.1 Known Attacks Detection

As mentioned in Section 3.3, there are 38 papers on IDSs focusing on known attack detection. In this section, we analyse these papers and discuss the existing methodologies used to detect or classify known attacks in in-vehicle networks. Known attacks are previously identified and documented, with patterns included in the training dataset. An IDS can reliably detect them because their signatures and features have already been learned. Detection of known attacks typically relies on supervised learning, where models are trained on labelled data. The section is organised into three subsections based on the features used to build the model: ID-based detection, payload-based detection, and CAN frame-based detection. Each subsection examines different approaches for identifying malicious activities, emphasizing their strengths and limitations. Figure 3.5 illustrates previous work on detecting known attacks, showing that most studies utilised a DL approach and used CAN frames as input features.

ID-Based Detection

Attacks such as injecting or deleting frames alter certain properties of message ID sequences compared to normal messages. This section presents research where the authors utilised these properties and used CAN IDs solely as an input feature to develop IDSs for detecting known attacks.

Song et al. [85] utilised the sequential behaviour of CAN data to identify message injection attacks. During these attacks, frequent frame injections resulted in distinct ID pattern changes, which were leveraged for detection. The authors relied solely on the bit-wise CAN ID sequence, which was processed directly as input, eliminating the need for additional feature engineering. They introduced a Deep Convolutional Neural Network (DCNN) model that was redesigned by minimizing unnecessary complexities within the Inception-ResNet architecture to achieve an optimised input size of $(29 \times 29 \times 1)$ and a binary classification output.

Refat et al. [86] used graph-based techniques to extract features from the CAN IDs in in-vehicle networks. The authors converted a window of CAN IDs into a graph and extracted seven graph properties, including the number of nodes, number of edges, radius, diameter, density, reciprocity, average clustering coefficient, and assortative coefficient, to use as input features. These extracted features were then used to train two traditional ML algorithms: Support Vector Machine (SVM) and K-nearest Neighbours (KNN) models. The experimental

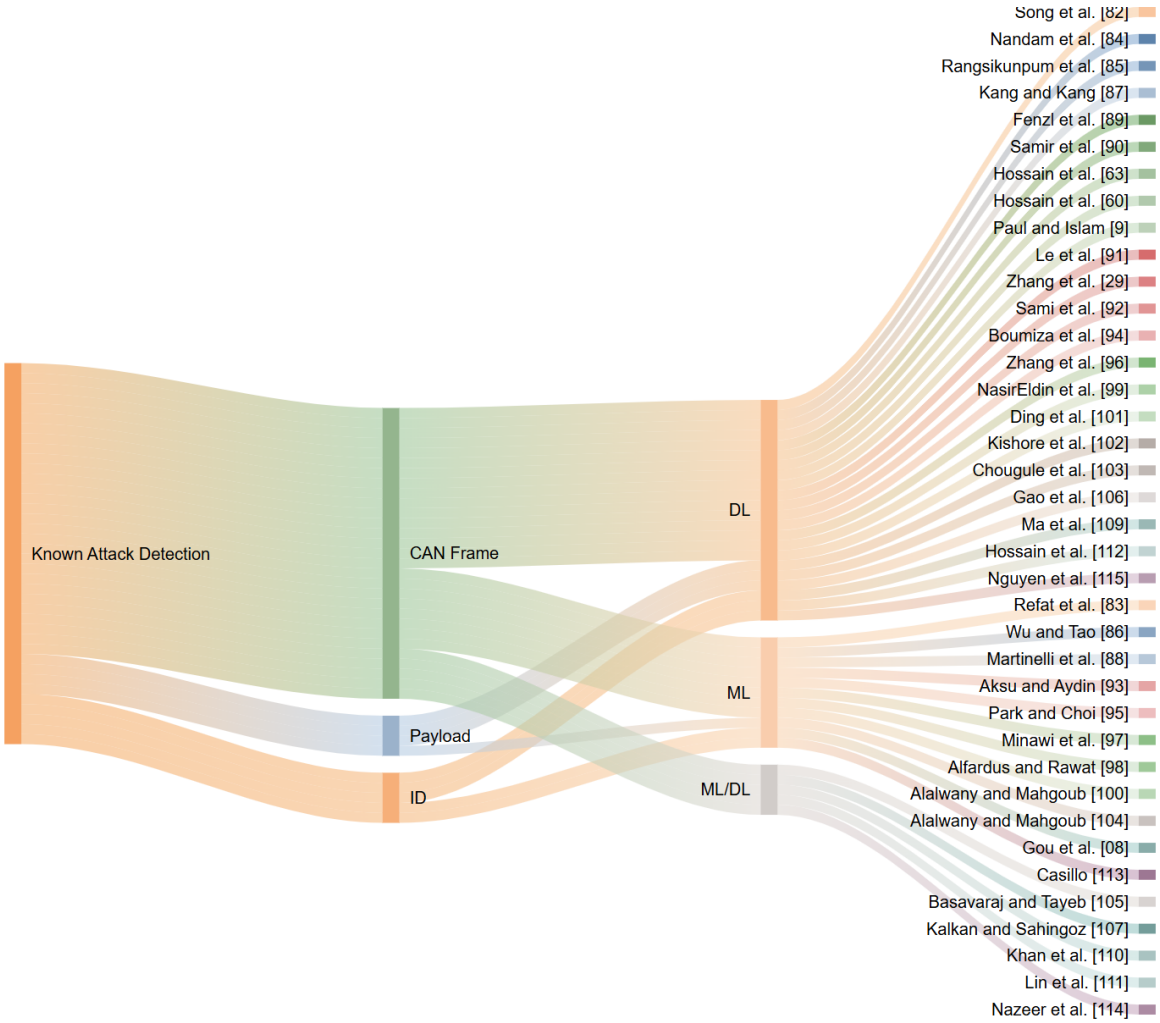


Fig. 3.5 Related work on known attack detection.

results demonstrated that using graph-based features outperformed the traditional CAN bus features.

Nandam et al. [87] employed a Long Short-Term Memory (LSTM) model to detect DoS attacks on the CAN bus. The model utilises the CAN ID of incoming messages to identify potential DoS attacks. A sequence of previous messages is stored and combined with the current message to form the input, enabling the model to predict and detect DoS attacks effectively.

Rangsikunpum et al. [88] introduced a Binarised Neural Network (BNN)-based IDS to identify attacks on the CAN bus. The primary objective of the proposed IDS is to deploy an ML model on a low-cost Field Programmable Gate Array (FPGA) device, optimising for low power consumption, minimal execution time, and high accuracy. By leveraging a 1-bit BNN model, the implementation is resource-efficient, making it suitable for deployment on cost-effective FPGA devices with reduced power requirements. Moreover, the IDS employs a two-stage architecture: the first stage identifies the presence of an attack, and the second stage, triggered only upon detecting an attack, performs detailed attack classification.

Wu and Tao [89] proposed a model based on ensemble learning using a Stacking integration approach. The method incorporates a meta-classifier composed of DTs, Extra Trees (ET), and Extreme Gradient Boosting (XGBoost). Final classification predictions are made by linearly combining input features and weights through a SoftMax meta-learner. Ensemble learning in this approach utilises the prediction results as new features, along with the true labels, to train the meta-learner.

Payload-Based Detection

Some attacks, such as spoofing, use legitimate CAN IDs but modify the CAN payload values, depending on the characteristics of the particular attack. These changes alter the pattern of payload sequences. This section discusses IDSs that utilise this property and use the CAN payload as an input feature to detect known attacks.

Kang and Kang [90] developed a Deep Neural Network (DNN)-based IDS to defend against malicious attacks. The authors utilised the 8-byte CAN payload to extract features, employing mode and value information to achieve dimensionality reduction. The initial weights for the DNN model were obtained from a separate Deep Belief Network. They then applied a template-matching method to compare the training samples with new CAN packets for detecting malicious messages. Although the proposed model demonstrated improved detection performance, it relied on mode and value information from CAN data, which presents significant challenges, especially without access to the DBC file.

Similarly, Martinelli et al. [91] utilised the eight CAN payload features to assess whether these features can effectively discriminate between attacks and normal messages. To address this question, they employed four fuzzy classification algorithms to identify four types of attacks targeting the CAN bus, including DoS, fuzzy, RPM, and gear spoofing. These algorithms include two types of fuzzy-rough KNN, the discernibility classifier, and a fuzzy unordered rule induction algorithm. The classification analysis was performed using the Weka3 tool. Experimental results indicated that the feature vector is a potential candidate for accurately classifying between injected and normal messages.

Fenzl et al. [92] used Decision Trees (DTs) trained through genetic programming (GP) to detect intrusions in the CAN bus. Their approach focuses solely on message payloads, with models trained individually for each CAN ID within the CAN bus training data. The authors compared their method with Artificial Neural Networks (ANNs). Experimental results showed that for most intrusions, the accuracy of the ANN was slightly higher, and the ANN had a significantly lower training time; however, the proposed GP method demonstrated significantly improved detection time.

Samir et al. [93] investigated two DL-based IDSs: one leveraging LSTM and the other utilising a one-dimensional Convolutional Neural Network (CNN). These two supervised learning algorithms serve as classifiers capable of categorising attacks into different types. The authors employed two public datasets and a new dataset that they manually generated using the ICSim simulation tool to cover more complex scenarios and attack types. Experimental results show that the LSTM-based IDS outperforms the CNN-based IDS, leveraging its ability to capture temporal patterns for robust detection of diverse CAN bus attacks.

CAN Frame-Based Detection

Rather than relying solely on CAN IDs or CAN payload as features, IDSs in the literature have utilised a combination of features to identify pattern changes in CAN data sequences. This approach offers the advantage of detecting alterations in CAN IDs and manipulations of the payload. This section reviews IDSs that use CAN IDs and payload as input features to detect known attacks. Some studies also incorporate the DLC feature or time differences between consecutive CAN IDs, in combination with CAN ID and payload.

Hossain et al. [66] proposed an LSTM-based IDS. The model considers both CAN ID, DLC, and payload as input features to detect point and contextual anomalies, where contextual anomalies arise from abnormal CAN ID sequences [27]. The proposed LSTM is trained on both benign and attack data and employs both binary and multi-class classification. The authors collected CAN messages from an actual Toyota hybrid car and generated three

attack scenarios, including DoS, fuzzy, and spoofing attacks. They compared the performance of the proposed LSTM method with the survival analysis method and found that the LSTM model achieves a higher detection rate than the other methods.

Following their earlier research, Hossain et al. [65] introduced a 1D CNN model as an alternative to the LSTM model proposed in their previous study. They collected normal datasets from three cars: Toyota, Subaru, and Suzuki, and injected anomalous frames to create attacks, including DoS, fuzzy, RPM, and gear spoofing. The proposed model achieved a high attack detection rate for all types of attacks. However, they considered fuzzy to be the most critical attack in the in-vehicle system, as it is difficult to detect due to its similarity to legitimate traffic within the CAN bus network.

Similarly, Paul and Islam [9] proposed an ANN-based anomaly detection method to identify unauthorised messages in the CAN bus. They utilised benign and attack classes from DoS and fuzzy datasets to train their ANN model. The model demonstrated a high detection accuracy in distinguishing between legitimate and anomalous messages, achieving nearly negligible rates of false positives and false negatives.

Le et al. [94] proposed an IDS for multiclass classification based on a combination of AE models and a time-embedded transformer. The AE-based packet-level extraction model learns a compressed representation of each CAN frame within a CAN sequence, while the time-embedded transformer, which replaces positional encoding with a timestamp encoding component, is used as the sequence extraction component.

Zhang et al. [31] introduced a Binarized CNN (BCNN)-based IDS, designed to leverage the temporal and spatial characteristics of CAN messages. The proposed IDS consists of two main components: an input generator and a BCNN model. The input generator converts CAN messages from feature vectors into image form, enabling the BCNN model to capture their temporal and spatial features. The second component employs the BCNN model to process the output images from the input generator. Experimental results demonstrated that the BCNN model is four times faster and requires less memory compared to a 32-bit CNN-based IDS.

Sami et al. [95] introduced the Network Embedded System Laboratory's IDS (NESLIDS), which employs a supervised DL algorithm based on a DNN. NESLIDS is designed as an anomaly detection system to identify three known attacks.

Aksu and Aydin [96] proposed a meta-heuristic algorithm, the Modified Genetic Algorithm (MGA), to select a subset of features by removing irrelevant ones, thereby improving classification performance and reducing dimensionality. They evaluated the effectiveness of the feature selection process using five classifiers: Support Vector Classifier (SVC), Logistic

Regression Classifier (LRC), Decision Tree Classifier (DTC), k-Nearest Neighbors Classifier (KNC), and Linear Discriminant Analysis Classifier (LDAC).

Boumiza et al. [97] proposed an IDS for the CAN bus based on a Multi-Layer Perceptron (MLP) neural network. The IDS first partitions data by the ID field of CAN packets, using the K-means clustering algorithm to create subclusters. It then extracts mode and frequency features from each subcluster to train the neural network. The proposed IDS operates separately for each CAN ID, combining the individual decisions to calculate a final score and trigger an alert in the event of an attack detection.

Park and Choi [98] proposed a Multi-labeled Hierarchical Classification (MLHC) IDS to detect message injection attacks. MLHC identifies the occurrence of attacks and classifies them using only pre-existing labeled attack data. The authors evaluated the method's performance using four ML algorithms: SGD, kNN, DT, and RF. Simulation results showed that the MLHC model achieved high accuracy with the RF algorithm and rapid detection with the DT algorithm.

Zhang et al. [99] proposed a Convolutional Encoder Network (CEN) model designed to detect network intrusions in CAN networks. The architecture integrates an encoder for dimensionality reduction, a CNN to increase network depth, and Inception ResNet to optimise training time. Additionally, the authors introduced a Feature-based Sliding Window method to extract features from the CAN Data Field and CAN IDs. Experimental results highlight the effectiveness of the feature-based sliding window in improving detection performance.

Minawi et al. [100] proposed an ML-based IDS system comprising three layers: the CAN Message Input Layer, the Threat Detection Layer, and the Alert Layer. The Threat Detection Layer utilises ML algorithms such as Random Tree (RT), Random Forest (RF), Stochastic Gradient Descent (SGD) with hinge loss, and Naive Bayes (NB) to detect different types of attacks. Additionally, this layer is designed with multiple modules, each tailored to detect specific types of attacks.

Similarly, Alfardus and Rawat [101] used the same proposed IDS in [100] but with four different ML algorithms, including KNN, RF, SVM, and Multilayer Perceptron (MLP), to detect CAN bus attacks.

NasirEldin et al. [102] proposed an attention-based model to detect CAN bus intrusions. The model consists of an attention layer that assigns higher importance to the most prominent features by calculating attention scores between the input features and the target, followed by a self-attention layer to identify relationships between data elements. Experimental results demonstrate that the proposed model outperformed baseline models, including an LSTM.

Alalwany and Mahgoub [103] proposed an ML-based IDS for detecting attacks on the CAN bus using supervised ML models, including RF, DT, Gaussian Naïve Bayes (GaussianNB), Logistic Regression (LR), AdaBoost, KNN, XGBoost, and Gradient Boosting. To further enhance attack detection accuracy, the authors combined all supervised models using three ensemble methods: voting, stacking, and bagging. The ensemble learning strategy offers the advantage of enabling models with different capabilities to complement one another in the classification task. Compared to individual models, the ensemble classifiers outperformed the supervised classifiers, improving the effectiveness of the supervised ML models by leveraging diverse learning mechanisms to support one another.

Ding et al. [104] proposed an IDS based on a Bidirectional LSTM (Bi-LSTM) network with a sliding window strategy. A two-dimensional input data sample set was constructed using the sliding window, and the Bi-LSTM network was trained on these features to learn a classifier for intrusion detection. Experimental results demonstrate that the proposed model outperforms other network models, except for DoS attacks.

Similarly, Kishore et al. [105] proposed a Bi-LSTM, which processes input data in both forward and backward orientations to detect anomalies in the CAN bus.

Chougule et al. [106] proposed HybridSecNet, a hybrid two-step LSTM-CNN IDS designed to enhance in-vehicle security. HybridSecNet consists of two classification stages: the first stage uses an LSTM to classify input data as either normal or attacked. If an attack is detected in the initial stage, the second stage is activated, employing a CNN-based multiclass classifier to identify and categorise the specific type of attack.

Moreover, Alalwany and Mahgoub [107] proposed an in-vehicle IDS to improve the accurate detection and classification of CAN bus attacks in real time using ensemble techniques and the Kappa Architecture. The Kappa Architecture facilitates real-time attack detection, while ensemble learning combines multiple ML classifiers, including RF, DT, and XGBoost, to enhance detection accuracy. The study demonstrated that ensemble approaches, which integrate the strengths of multiple models, significantly improved detection accuracy and robustness.

Basavaraj and Tayeb [108] proposed a lightweight DNN-based model to detect and classify attacks on the CAN bus. The proposed model outperformed baseline models, including RF, DTs, and the kNN algorithm.

Gao et al. [109] proposed a CNN and Bi-LSTM model with multi-head attention for attack detection and classification. The CNN module enhances feature extraction, the Bi-LSTM module captures sequential features and relationships, and the multi-head attention module identifies further correlations between features.

Kalkan and Sahingoz [110] applied six different ML models: RF, bagging, ADA boosting, NB, LR, and ANN. Their experimental results demonstrated that tree-based and ensemble learning algorithms achieved superior performance. However, the authors did not specify the features used for training, leading to the assumption that all features were included.

Gou et al. [111] proposed an Adaptive Tree-based Ensemble Network (ATBEN) as the intrusion detection engine for IDS in the Internet of Vehicles (IoV). ATBEN leverages a variety of ML models, including XGBoost, LightGBM, RF, and ET, as base estimators, stacking them into layers within the network. The cascading connections between layers facilitate precise and efficient multiclass classification. The authors demonstrated the effectiveness of the proposed IDS by evaluating its performance against a range of cyberattacks targeting both in-vehicle systems and external networks within the IoV.

Ma et al. [112] introduced a lightweight IDS for the CAN bus, leveraging a GRU-based architecture. To enhance efficiency, they employed a low-complexity feature extraction algorithm to derive features from CAN frames. The proposed model demonstrated near real-time performance and outperformed baseline models in detection accuracy.

Khan et al. [113] proposed DivaCAN, an IDS that combines DL models with conventional ML methods through an ensemble of base classifiers, including DNN, MLP, light gradient-boosting machines, ET, RF, Bagging, and KNN to detect intrusions on the CAN bus. To improve detection performance, a meta-classifier aggregates the outputs of the base classifiers in a weighted and adaptive manner, considering their performances and correlations. This work addresses the trade-off between false positives and time complexity in CAN bus IDS.

Lin et al. [114] proposed a CNN-based approach leveraging the VGG16 classifier to learn attack behaviour characteristics and classify attacks. Feature vectors were transformed into feature images, which were then input into the VGG16 model for accurate categorisation of cyberattacks in in-vehicle networks. To ensure high precision in predicting the stability of network intrusion detection, the approach combines the VGG16 model with the XGBoost ensemble learning algorithm, enabling effective analysis of suspicious network traffic.

Hossain et al. [115] proposed an LSTM-based IDS for detecting in-vehicle attacks. The CAN message data was collected using a tool called Vehicle Spy 3. To evaluate the IDS, the authors employed both binary and multi-class classification approaches, utilizing vanilla LSTM and stacked LSTM models. Since the dataset originally contained no attacks, the authors simulated DoS, Fuzzing, and Spoofing attacks on the CAN bus of a Toyota Hybrid car using a Python-based program.

Casillo [116] proposed an embedded IDS for automotive systems by adopting Bayesian Networks for the rapid identification of malicious messages on the CAN bus. The CAN bus

dataset was generated by simulating vehicle driving for approximately 24 hours on a city track within the CARLA environment. During the simulation, the vehicle was subjected to attacks to replicate potential intrusion scenarios based on specific use cases.

Nazeer et al. [117] proposed a hybrid approach, DeepXG, which combines the XGBoost and DNN models to detect and classify attacks on the CAN bus. The XGBoost model is trained on the dataset to extract critical features and reduce computational complexity, while the DNN leverages these learned representations to detect anomalies and intrusions.

Nguyen et al. [118] proposed an IDS based on a Transformer attention network for a CAN bus, designed to analyse a single message. The proposed IDS includes two models: one using only a single message and another leveraging sequential CAN IDs. The first model effectively detects DoS, fuzzy, and spoofing attacks but cannot detect replay attacks due to its reliance on single-message analysis. To address this limitation, the second model was designed to detect replay attacks by incorporating sequential CAN ID information. Additionally, the proposed model employs transfer learning to enhance the performance of models trained on small datasets from other car models.

Table 3.3 summarises the related work on known attack detection methods, including the learning approach, binary or multi-class classification, dataset used, detectable attacks, employed algorithm, and the model size or the size of trainable parameters. In Table 3.3, we assume that papers that do not explicitly state the input features used are referring to CAN frame features. Among these studies, only three [85, 92, 94] measure the trainable parameters, reflecting the model's size, while the others did not consider model size for deployment.

Table 3.3 Summary of related work on known attack detection methods.

Reference	Year	ML / DL	Category	Classification Type	Dataset	ID	Payload	Attack Types	Algorithm	Model Size/ Trainable Parameters
ID-Based Attack Detection										
[85]	2020	DL	Supervised	Binary	Car Hacking [1]	✓		Message Injection	DCNN	1.76 Million
[86]	2022	ML	Supervised	Binary	Car Hacking [1]	✓		DoS, Fuzzy, RPM Spoofing	SVM, KNN	N/A
[87]	2022	DL	Supervised	Binary	car-hacking [85]	✓		DoS	LSTM	N/A
[88]	2024	DL	Supervised	Binary/ Multi-class	Car Hacking [1]	✓		(Gear, RPM) Spoofing, DoS, Fuzzy	BNN	4.85 Mb
[89]	2024	ML	Supervised	Binary	Car Hacking [1]	✓		(Gear, RPM) Spoofing, DoS, Fuzzy	Ensemble model (DT, ET, XGBoost)	N/A
Payload-Based Attack Detection										
[90]	2016	DL	Supervised	Binary	Simulation		✓	Injection Attacks	DNN	N/A
[91]	2017	ML	Supervised	Binary	Car Hacking [1]		✓	DoS, Fuzzy, (Gear, RPM) Spoofing	KNN	N/A
[92]	2021	ML	Supervised	Binary	Car Hacking [1], Tesla Model X, Renault Zoe electric car		✓	(RPM, Gear) Spoofing	DT, GP	1,101

Continued on next page

Table 3.3 (continued): Summary of related work on known attack detection methods.

Reference	Year	ML / DL	Category	Classification Type	Dataset	ID	Payload	Attack Types	Algorithm	Model Size/ Trainable Parameters
[93]	2024	DL	Supervised	Multi-class	Car Hacking[1], OTIDS [61], Own		✓	DoS, Fuzzy, Spoofing, Replay	CNN, LSTM	N/A
CAN Frame -Based Attack Detection										
[66]	2020	DL	Supervised	Binary/ Multi-class	Own	✓	✓	DoS, Fuzzy, Spoofing	LSTM	N/A
[65]	2020	DL	Supervised	Binary/ Multi-class	Own	✓	✓	DoS, Fuzzy, Spoofing	CNN	N/A
[9]	2021	DL	Supervised	Binary	OTIDS [61]	✓	✓	DoS, Fuzzy	ANN	N/A
[94]	2024	DL	Supervised	Multi-class	car-hacking [85], ROAD [119]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy, Fabrication, Masquerade	AE, Time-embedded Transformer	259,000
[31]	2024	DL	Supervised	Binary	Own	✓	✓	Replay, Spoofing	BCNN	N/A
[95]	2020	DL	Supervised	Binary	OTIDS [61], ML350 [120]	✓	✓	DoS, Fuzzy, Impersonation	DNN	N/A
[96]	2022	ML	Supervised	Binary/ Multi-class	car-hacking [85]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy	SVC, LRC, DTC, KNC, LDAC	N/A

Continued on next page

Table 3.3 (continued): Summary of related work on known attack detection methods.

Reference	Year	ML / DL	Category	Classification Type	Dataset	ID	Payload	Attack Types	Algorithm	Model Size/ Trainable Parameters
[97]	2019	DL	Supervised	Binary	Dataset [121]	✓	✓	Frequency modification, Data-content modification	MLP	N/A
[98]	2020	ML	Supervised	Binary/ Multi-class	Survival Analysis Dataset [122]	✓	✓	Fuzzy, Flooding, Malfunction	SGD, kNN, DT, RF	N/A
[99]	2020	DL	Supervised	Multi-class	Car Hacking [1], car-hacking [85]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy	CEN	N/A
[100]	2020	ML	Supervised	Binary	Car Hacking [1]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy	RT, RF, SGD, NB	N/A
[101]	2021	ML	Supervised	Binary	Car Hacking [1]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy	KNN, RF, SVM, MLP	N/A
[102]	2021	DL	Supervised	Binary	Car Hacking [1]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy	Attention-based model	N/A

Continued on next page

Table 3.3 (continued): Summary of related work on known attack detection methods.

Reference	Year	ML / DL	Category	Classification Type	Dataset	ID	Payload	Attack Types	Algorithm	Model Size/ Trainable Parameters
[103]	2022	ML	Supervised	Binary	Car Hacking: Attack & Defence Challenge 2020 [2]	✓	✓	Flooding, Spoofing, Replay, Fuzzy	LR, k-NN, RF, GaussianNB, Gradient Boosting, AdaBoost, DT, XGBoost	N/A
[104]	2022	DL	Supervised	Binary	Car Hacking [1]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy	Bi-LSTM	N/A
[105]	2024	DL	Supervised	Binary	Car Hacking: Attack & Defence Challenge 2020 [2]	✓	✓	Flooding, Spoofing, Replay, Fuzzy	Bi-LSTM	N/A
[106]	2024	DL	Supervised	Binary/ Multi-class	Car Hacking [1]	✓	✓	DoS, Fuzzy, (Gear, RPM) Spoofing	LSTM-CNN	N/A
[107]	2024	ML	Supervised	Multi-class	Car Hacking: Attack & Defence Challenge 2020 [2]	✓	✓	DoS, Spoofing, Replay, Fuzzy	RF, DT, XGBoost	N/A
[108]	2022	DL / ML	Supervised	Multi-class	CAN dataset [123]	✓	✓	Reconnaissance, DoS, Fuzzy	DNN	N/A
[109]	2023	DL	Supervised	Multi-class	Car Hacking [1]	✓	✓	DoS, Fuzzy, (Gear, RPM) Spoofing	CNN, bi_LSTM	N/A

Continued on next page

Table 3.3 (continued): Summary of related work on known attack detection methods.

Reference	Year	ML / DL	Category	Classification Type	Dataset	ID	Payload	Attack Types	Algorithm	Model Size/ Trainable Parameters
[110]	2020	ML / DL	Supervised	Binary	Car Hacking [1]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy	RF, bagging, ADA boosting, NB, LR, ANN	N/A
[111]	2023	ML	Supervised	Multi-class	car-hacking [85]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy	XGBoost, LightGBM, RF, ET	N/A
[112]	2022	DL	Supervised	Binary	Car Hacking [1]	✓	✓	DoS, Spoofing, Fuzzy	GRU	N/A
[113]	2024	ML / DL	Supervised	Multi-class	OTIDS [61]	✓	✓	DoS, Fuzzy, Impersonation	DNN, MLP, light gradient-boosting machine, ET, RF, Bagging, KNN	N/A
[114]	2022	DL / ML	Supervised	Multi-class	Car Hacking [1]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy	VGG16, XGBoost	N/A
[115]	2020	DL	Supervised	Binary/ Multi-class	Own	✓	✓	DoS, Fuzzy, Spoofing	LSTM	N/A
[116]	2019	ML	Supervised	Binary	Simulation	✓	✓	Turn right, Turn left, Brake	Bayesian Network	N/A

Continued on next page

Table 3.3 (continued): Summary of related work on known attack detection methods.

Reference	Year	ML / DL	Category	Classification Type	Dataset	ID	Payload	Attack Types	Algorithm	Model Size/ Trainable Parameters
[117]	2024	ML / DL	Supervised	Multi-class	Own	✓	✓	Flooding, Replay, Spoofing	XGBoost, DNN	N/A
[118]	2023	DL	Supervised	Binary/ Multi-class	Car Hacking [1], IVN [124], Survival Analysis Dataset [122]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy, Replay, Malfunction	Transformer	N/A

Limitations of Known Attacks Detection

Although existing known attack detection approaches often report high detection accuracy and low False Alarm Rates (FAR), their effectiveness is strongly dependent on the availability of well-labelled and balanced attack datasets. The acquisition of such datasets remains a major challenge in the in-vehicle security domain [70], and current studies largely overlook the practical difficulties of the labelling process, which is time-consuming, error-prone, and tedious [70, 125]. This reveals a clear research gap in the literature, as existing known attack detection approaches fail to consider how detection performance can be maintained in the presence of limited or imperfectly labelled data. Moreover, existing studies predominantly focus on recognising predefined attack patterns and offer limited capability to handle attack deviations or previously unknown attacks. As attackers continuously adapt their strategies to evade detection, supervised learning based models struggle to identify unseen or modified attacks that are not represented in the training data [20, 28]. This exposes a fundamental gap in current known attack detection research, where existing approaches fail to provide mechanisms that ensure robustness against evolving, unseen, or modified attacks beyond static and predefined attack representations.

3.4.2 Unknown Attacks Detection

As mentioned in Section 3.3, there are 27 papers on IDSs focusing on unknown attack detection or anomaly detection. In this section, we analyse these papers, discuss the existing methodologies used to detect new, unknown attacks in in-vehicle networks and identify research gaps. Unknown attacks are new patterns not stored in databases and unseen by the model during training, representing entirely new classes beyond the classifier's prior experience [126]. Detection of unknown attacks typically relies on unsupervised learning, where models are trained solely on normal data, relying on profiling normal traffic behaviours 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 [127]. The section is organised into three subsections based on the features used to build the model: ID-based detection, payload-based detection, and CAN frame-based detection. Each subsection examines different approaches for identifying malicious activities, emphasizing their strengths and limitations. Figure 3.6 illustrates previous work on detecting unknown attacks. As depicted in the figure, similar to known attack detection, most studies on unknown attack detection utilised a DL approach and used CAN frames as input features.

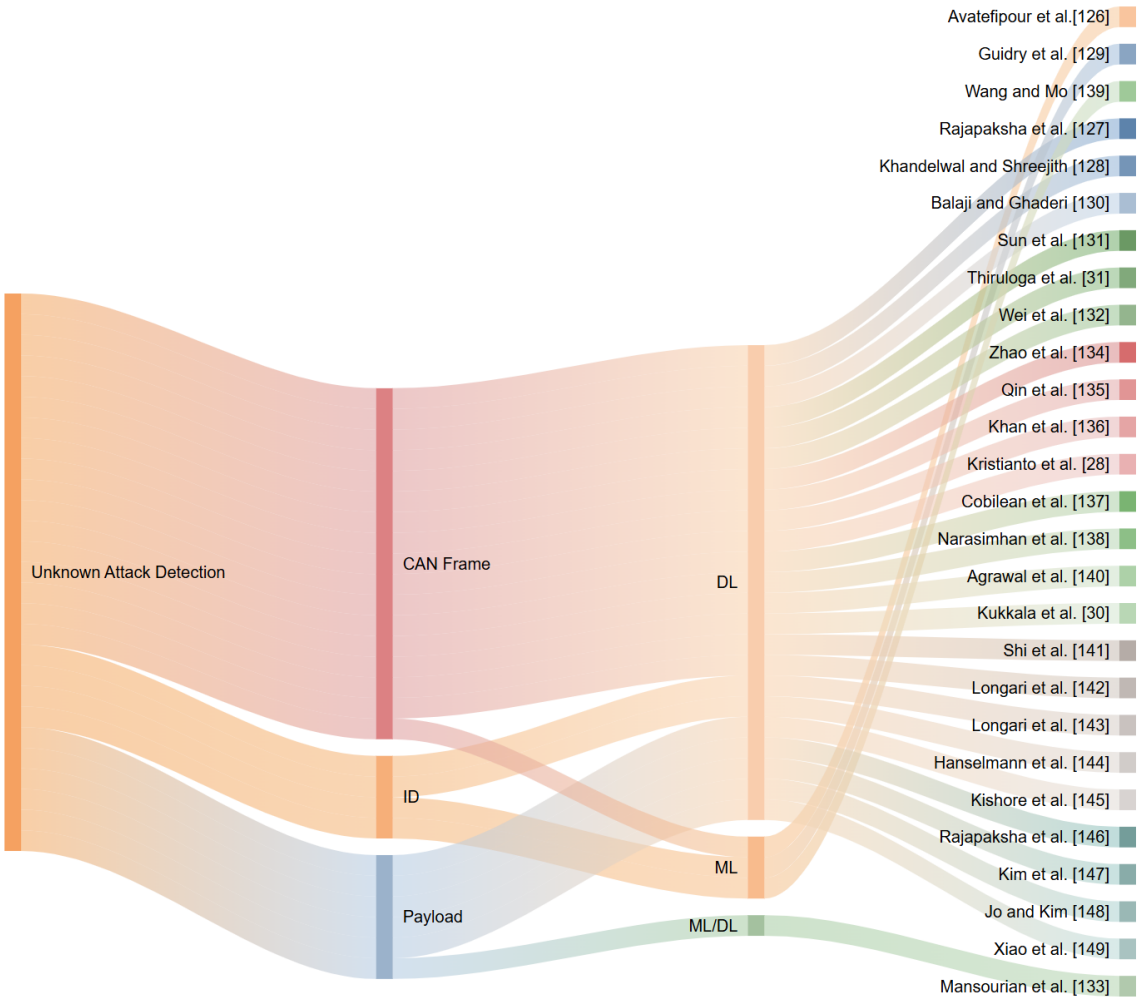


Fig. 3.6 Related work on unknown attack detection.

ID-Based Detection

This section reviews research where authors used only CAN IDs as the input feature to develop IDSs for detecting new, unknown attacks.

Avatefipour et al. [128] proposed an anomaly detection model based on a modified One-class Support Vector Machine (OCSVM) incorporating the Modified Bat Algorithm (MBA). The model was built using normal CAN bus traffic, which exhibits recurring patterns in CAN IDs under normal conditions. Any deviation from this normal traffic, such as increased message occurrence frequency or message flooding, is detected by the model as malicious activity. For evaluation, CAN bus data were collected from a licensed, unmodified vehicle and two public CAN bus datasets. The authors compared the proposed model with baseline Isolation Forest and classical OCSVM models, finding that the MBA-OCSVM achieved the highest true positive rate and the lowest false alarm rate compared to both alternatives.

Rajapaksha et al. [129] proposed CAN-CID, a context-aware IDS aimed at addressing the computational inefficiency of N-gram-based models while detecting a wide range of cyberattacks on the CAN bus. CAN-CID utilises an ensemble approach that combines a Gated Recurrent Unit (GRU) network and a time-based model. The single-layer GRU network detects anomalous ID sequences and minimises detection latency, while the time-based model identifies anomalies using time-based thresholds. The anomaly-to-total-ID ratio within an observation window is then used to classify the window as either anomalous or benign. This study highlights the effectiveness of ensemble models in detecting diverse attacks on the CAN bus.

Khandelwal and Shreejith [130] presented a Convolutional Autoencoder (CAE) model for detecting zero-day attacks, trained solely on benign CAN messages. Leveraging Vitis-AI tools, they quantised the model to optimise performance on resource-constrained platforms. The proposed IDS achieves state-of-the-art classification accuracy across multiple unseen attacks, along with a 1.3x speed-up in processing latency and approximately 2x reduction in power consumption compared to existing state-of-the-art IDSs.

Guidry et al. [131] proposed the use of the OC-SVM to detect anomalous data on a vehicle's CAN bus. Instead of utilising raw CAN bus data, three distinct features were extracted for each unique CAN ID: the average frequency of appearance of a CAN ID, the average time interval between consecutive appearances of a CAN ID, and the standard deviation of transmission times for CAN IDs. These features were selected because they rely on the temporal and behavioural characteristics of message transmissions rather than the data content within the messages. The model was trained on CAN bus data collected

under normal operating conditions, making it well-suited for detecting unknown attacks in vehicular networks.

Payload-Based Detection

This section discusses IDSs that use the 8-byte CAN payload as an input feature to identify new, unknown attacks.

Balaji and Ghaderi [132] proposed NeuroCAN, a contextual anomaly detection model that consists of an embedding layer and LSTM to learn the spatio-temporal correlations among CAN payload values. The embedding layer performs a linear transformation of the input data from each CAN ID, passes it through a sigmoid function, and accumulates it over all IDs. This is followed by an LSTM and an output layer, forming a prediction-based anomaly detector. The use of payload values from other IDs as context enables the capture of inter-ID correlations. However, the model is trained separately for each CAN ID, resulting in high memory and computational costs.

Sun et al. [133] proposed a CNN-LSTM-based IDS with an attention mechanism. The model used one-dimensional convolution to extract abstract features, while a bi-directional LSTM was employed to capture time dependencies. The bit flip rate was used to identify continuous fields from the 64-bit payload, resulting in a 41-bit smaller signal, which is more efficient than directly predicting the full 64-bit. Experiments demonstrated that this approach reduces data dimensionality and improves model training efficiency. The pre-processed data was fed into the neural network model to predict the output signal and determine whether the received signal was abnormal. The proposed model improved attack detection accuracy by 2.5% compared to related research.

Thiruloga et al. [33] introduced TENET, a novel anomaly detection framework based on Temporal Convolutional Neural Attention (TCNA) networks. TENET takes a sequence of signal values from a message as input and uses CNNs to predict the signal values of the next message instance by learning the underlying probability distribution of normal data. A DT-based classifier was then employed as the attack detector. Experimental results showed that TENET achieved a 3.32% improvement in detection accuracy, a 32.7% reduction in the false negative rate, and 94.62% fewer model parameters compared to a baseline model. However, the model processed data ID-wise, training separate models for each ID, which limits its ability to detect anomalies, such as collective anomalies, that arise from interactions between different CAN IDs.

Wei et al. [134] introduced AMAEID, a multi-layer denoising autoencoder model. The model takes only the 8-byte payload of the CAN message as input. It first transforms the raw

hexadecimal payload into binary format, then applies a multi-layer denoising autoencoder to extract deeper hidden features that represent the underlying characteristics of the message. Additionally, AMAEID utilises an attention mechanism and a fully connected layer to classify messages as normal or abnormal. Experimental results demonstrate that AMAEID surpasses traditional ML algorithms like DT, KNN, and LinearSVC. However, the model was trained and tested using only two CAN IDs.

Mansourian et al. [135] proposed an anomaly-based IDS to detect attacks on the in-vehicle CAN bus. The proposed IDS comprises three modules: an LSTM model, a prediction error calculator, and a Gaussian Naïve Bayes (GNB) classifier. The LSTM is trained on normal CAN messages to learn the typical sequential behaviour of each ECU. Once trained, the network predicts the next expected payload of an ECU based on past observations and compares it to the actual received value. When an attack occurs, the trained LSTM network fails to make accurate predictions, resulting in higher-than-normal prediction errors. The GNB classifier then classifies messages as either normal or an attack based on these prediction errors.

Zhao et al. [136] introduced the Same Origin Method Execution (SOME) attack, which mimics the period, clock skew, and voltage of normal messages, making detection by existing IDSs challenging. To address this, they developed a GAN-based IDS, named GVIDS, which employs one-hot encoding to represent data and converts data frames into CAN images. This approach is effective as attacks either directly alter frame data or disrupt frame sequences, indirectly modifying all the consecutive data fields. Experiments on two real vehicles show that GVIDS successfully detects SOME attacks as well as other existing attack types.

CAN Frame-Based Detection

This section reviews IDSs that use the CAN frame (CAN IDs and payload) as input features to detect new, unknown attacks. Some studies also incorporate the DLC feature and/or time differences between consecutive CAN IDs, in combination with CAN ID and payload.

Qin et al. [137] proposed an LSTM-based anomaly detection algorithm to detect abnormal behaviour on the CAN bus. CAN data were collected from the network of a real vehicle, with simulated attacks such as tampering or inserting duplicate random packets into the CAN bus. CAN ID and payload were converted from hexadecimal to binary representations instead of decimal, which increased the dimensionality of the features. Anomaly detection in the message stream of the CAN bus was performed for each ID separately. Experimental results showed that the proposed model could detect anomalous data with over 90% accuracy.

Khan et al. [138] used a bidirectional LSTM model with an improved feature processing technique to address the challenge of zero-day attacks. The proposed IDS is a multi-stage system, where the initial stage employs a state-based Bloom filter technique to verify the states of incoming data, while the second stage uses a bidirectional LSTM classifier to detect cyberattacks. They applied enhanced data pre-processing to improve the scalability and performance efficiency of the IDS, including feature conversion, feature reduction, and feature normalization. Principal component analysis was used for feature reduction. Experimental results showed that the feature pre-processing led to a 19.31% improvement in accuracy compared to raw data.

Kristianto et al. [30] proposed a lightweight unsupervised IDS on a simple Recurrent Neural Network (RNN). The authors suggest deploying the IDS model at each domain gateway, leveraging the computational resources of the gateways to handle only domain-specific messages. This approach enables the gateway to be optimised for detecting malicious messages within its domain while maintaining a lightweight design. The IDS achieves up to a 94% reduction in parameters compared to existing models, significantly decreasing memory usage and energy consumption. Despite this reduction in size, the proposed models demonstrate only a slight decrease in accuracy compared to current solutions.

Cobilean et al. [139] proposed a Transformer neural network-based IDS designed to predict anomalous behaviour within CAN protocol communication. The Transformer model is trained to predict the next communication sequence, and anomalies are detected when the difference between the predicted sequence and the actual received sequence exceeds a defined threshold. A key advantage of this model is that it does not require labelled attack data for learning the communication sequence.

Narasimhan et al. [140] proposed an unsupervised two-stage approach that combines DL with a probabilistic model for anomaly detection. In the first stage, an Autoencoder (AE) is used to extract optimal features that differentiate between normal data and attacks on the CAN bus. Unlike other autoencoder-based models that utilise the reconstructed signal for anomaly detection, this model leverages the latent space as input to a Gaussian Mixture Model (GMM). In the second stage, the GMM clusters these features into normal and attack categories. Experimental results demonstrated that the proposed method achieved superior performance across various datasets. For evaluation, a real dataset from a Mercedes ML350 was used; however, as this dataset contained only four CAN IDs, the practical applicability of the model may be constrained.

Wang and Mo [141] proposed a CAN bus anomaly detection model based on the FLXG-Boost algorithm. To address the challenge posed by the large volume of traffic data messages

with limited features, they introduced a newly defined feature: information entropy, which serves as an additional set of features in the CAN message data domain.

Agrawal et al. [142] proposed NovelADS, an IDS that utilises CNNs and LSTMs to detect anomalies in CAN network traffic. NovelADS captures spatio-temporal features and long-term dependencies from CAN messages. The DL models are trained on normal CAN data, and the system classifies incoming CAN data as genuine or anomalous using a reconstruction-based thresholding approach.

Kukkala et al. [32] proposed INDRA, an IDS based on a GRU-based recurrent autoencoder designed to learn latent representations of normal CAN traffic and detect anomalies on the CAN bus. At runtime, the trained autoencoder monitors deviations from normal behaviour to identify potential intrusions. Signal-level intrusion scores, calculated as the difference between predicted and actual signal values, are used to identify anomalous signals. The authors trained separate autoencoder models for each CAN ID, enabling ID-specific anomaly detection model.

Shi et al. [143] introduced an IDS called IDS-DEC, which integrates a spatiotemporal self-encoder employing LSTM and CNN (LCAE) with an entropy-based deep embedding clustering approach. The LSTM component models the sequential nature of the data, capturing long-term dependencies in the time-series data from the CAN bus. Additionally, as network data can be represented as a multidimensional matrix with spatial structure, CNNs are employed to extract key features, thereby enhancing the accuracy and efficiency of detection. Experimental results demonstrate that the proposed IDS achieves superior detection performance compared to traditional ML algorithms and other deep clustering methods.

Longari et al. [144] introduced CANnolo, an IDS based on LSTM autoencoders for identifying anomalies on the CAN bus. CANnolo analyses CAN message streams to construct a model of normal data sequences and detects anomalies by measuring the discrepancy between reconstructed sequences and their corresponding real sequences. The authors partitioned the dataset into groups based on CAN IDs, with each group processed independently and trained on separate models. While this approach simplifies the training process, it limits the system's ability to detect signal correlations, thereby reducing its effectiveness in identifying anomalies such as collective anomalies [27].

To improve the overall architecture and reduce the computational requirements of CANnolo, ensuring it meets the real-time constraints of the automotive domain, Longari et al. [145] then proposed CANdito, an unsupervised IDS that leverages LSTM autoencoders to detect anomalies using a signal reconstruction process. CANdito reconstructs the time series of CAN packets for each ID and calculates anomaly scores based on the reconstruction error.

Hanselmann et al. [146] proposed CANet, an LSTM-based autoencoder designed to identify attacks on the CAN bus. Separate LSTM models were used for each CAN ID, with their outputs concatenated into a single latent vector. The difference between the original and reconstructed signal values was utilised to determine the normal status. Experimental results showed that the model achieved a high detection rate with low false-positive and false-negative rates across various attack types.

Similarly, Kishore et al. [147] proposed an LSTM-based anomaly detection method. The model outperforms previous tree-based ML algorithms, including AdaBoost, GBoost, Bagging, XGBoost, and LGBM.

Rajapaksha et al. [148] introduced an ensemble IDS that combines a GRU network and a novel AE model called Latent AE to identify cyberattacks on the CAN bus. The GRU network analyses the CAN ID field, while Latent AE focuses on the CAN payload field to identify anomalies. To improve efficiency, Latent AE incorporates Cramér's statistic-based feature selection and a transformed CAN payload structure. By utilising a compact latent space, it overcomes the issue of high false negatives in traditional AEs caused by overgeneralisation. Experimental findings reveal that the ensemble IDS enhances attack detection and addresses the limitations of the individual models.

Kim et al. [149] proposed an IDS based on multiple LSTM-Autoencoders that utilise diverse features, including transmission intervals and payload value changes, to capture various characteristics of normal network behaviour. The system consists of a feature sequence extractor, LSTM-Autoencoder models, and an anomaly detector. The time interval sequence extractor calculates the intervals between consecutive frames with the same ID, generating a chronological sequence for each ID. Similarly, the Hamming distance sequence extractor computes the Hamming distances between the payloads of consecutive frames within ID-based streams. These feature sequences are processed by the LSTM-Autoencoders to produce reconstructed sequences. The anomaly detector evaluates the differences between the original time interval and Hamming distance sequences and their reconstructed counterparts, using these differences to determine whether the frame sequences are normal or anomalous.

Jo and Kim [150] proposed an IDS based on the Transformer architecture, which predicts the next data point based on the flow of previously input data. The IDS can detect attacks affecting both the temporal and spatial aspects of the data, as CAN data comprises temporal information recorded over time and spatial information recorded across devices. This two-dimensional data is used to train the model, achieving higher performance compared to using one-dimensional data.

Xiao et al. [151] proposed an anomaly detection IDS for in-vehicle networks based on a Convolutional LSTM Network (ConvLSTM), which accounts for both temporal and spatial correlations. The ConvLSTM model is first trained on benign CAN data, and its predictions are used to calculate the correlation coefficient with actual data. Abnormal behaviour is detected by comparing the correlation coefficients between the predicted and real data. Experimental results indicate that the ConvLSTM model maintains a stable correlation coefficient for normal data, while the coefficient for attack data declines rapidly over time. Compared to the LSTM model, the ConvLSTM model more effectively captures the underlying features of benign data, producing a more consistent correlation coefficient for attack-free states. Furthermore, the sharp drop in the correlation coefficient for attack data can facilitate the detection of unknown attacks.

Table 3.4 summarises the related work on unknown attack detection methods, including the learning approach, dataset used, detectable attacks, employed algorithm, and the model size or the size of trainable parameters. In Table 3.4, we assume that papers that do not explicitly state the input features used are referring to CAN frame features.

Among these studies, only three [85, 92, 94] measure the trainable parameters, reflecting the model's size, while the others did not consider model size for deployment.

Table 3.4 Summary of related work on unknown attack detection methods.

Reference	Year	ML / DL	Category	Dataset	ID	Payload	Attack Types	Algorithm	Model Size/ Trainable Parameters
ID-Based Attack Detection									
[128]	2019	ML	Unsupervised	Own, Dodge [152], OTIDS [61]	✓		Injection	MBA-OCSVM	N/A
[129]	2022	DL	Unsupervised	ROAD [119], car-hacking [85], Survival Analysis Dataset [122]	✓		Fabrication, Suspension, Masquerade	GRU	N/A
[130]	2023	DL	Unsupervised	car-hacking [85]	✓		DoS, Fuzzy, (Gear, RPM) Spoofing	AE	N/A
[131]	2023	ML	Unsupervised	Own	✓		Random ID, Zero ID, Replay	OC-SVM	N/A
Payload-Based Attack Detection									
[132]	2021	DL	Unsupervised	Two Public Datasets from [1]		✓	Flood, Replay, Drop, Spoofing, Fuzzy	LSTM	N/A

Continued on next page

Table 3.4 (continued): Summary of related work on unknown attack detection methods.

Reference	Year	ML / DL	Category	Dataset	ID	Payload	Attack Types	Algorithm	Model Size/ Trainable Parameters
[133]	2021	DL	Unsupervised	CAN Signal Extraction and Translation [153]		✓	Flood, Replay, Drop, Spoofing, Fuzzy	CNN-LSTM	682 KB
[33]	2022	DL	Unsupervised	Simulation		✓	Plateau, Continuous Change, Playback, Suppress	CNN	59.62 KB / 6064
[134]	2022	DL	Unsupervised	OTIDS [61]		✓	Payload value Manipulation	AE	N/A
[135]	2023	ML / DL	Unsupervised	Car Hacking [1], Survival Analysis Dataset [122]		✓	(Gear, RPM) Spoofing, DoS, Fuzzy, Flooding, Malfunction	LSTM, GNB	N/A
[136]	2022	DL	Unsupervised	Own		✓	Spoofing, Bus-off, Masquerade, SOME attacks	GAN	N/A
CAN Frame-Based Attack Detection									
[137]	2021	DL	Unsupervised	Own	✓	✓	Random CAN payload Values	LSTM	N/A

Continued on next page

Table 3.4 (continued): Summary of related work on unknown attack detection methods.

Reference	Year	ML / DL	Category	Dataset	ID	Payload	Attack Types	Algorithm	Model Size/ Trainable Parameters
[138]	2021	DL	Unsupervised	Car Hacking [1]	✓	✓	DoS, Fuzzy, RPM, Gear Spoofing	LSTM	N/A
[30]	2024	DL	Unsupervised	Car Hacking [1]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy	RNNs and AEs	Multiple IDSs (119 -272) KB for each gateway
[139]	2023	DL	Self-supervised	Survival Analysis Dataset [122]	✓	✓	Malfunction	Transformer	N/A
[140]	2021	DL	Unsupervised	ML350 [120]	✓	✓	DoS , Fuzzy	AE, GMM	N/A
[141]	2021	ML	Supervised	Simulation, OTIDS [61]	✓	✓	(Gear, RPM) Spoofing	FLXGBoost	N/A
[142]	2022	DL	Unsupervised	Car Hacking[1]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy	CNNs, LSTMs	N/A
[32]	2020	DL	Unsupervised	SynCAN [146]	✓	✓	Flooding, Plateau, Continuous, Suppress, Playback	GRU AE	443 kB

Continued on next page

Table 3.4 (continued): Summary of related work on unknown attack detection methods.

Reference	Year	ML / DL	Category	Dataset	ID	Payload	Attack Types	Algorithm	Model Size/ Trainable Parameters
[143]	2024	DL	Unsupervised	Car Hacking [1], Car Hacking: Attack & Defence Challenge 2020 [2]	✓	✓	(Gear, RPM) Spoofing, DoS, Replay, Fuzzy	LSTM, CNN, AE	N/A
[144]	2020	DL	Unsupervised	Recan [154]	✓	✓	Interleave, Discontinuity, Data field anomalies	LSTM -AE	<10 MB
[145]	2023	DL	Unsupervised	Recan [154], car-hacking [85]	✓	✓	(Gear, RPM) Spoofing, DoS, Fuzzy, Masquerade, Seamless change, Replay	LSTM AE	N/A
[146]	2020	DL	Unsupervised	SynCAN [146]	✓	✓	Flooding, Plateau, Continuous, Suppress, Playback	LSTM AE	N/A
[147]	2022	DL	Unsupervised	Car Hacking: Attack & Defence Challenge 2020 [2]	✓	✓	Flooding, Spoofing, Replay, Fuzzy	LSTM	N/A

Continued on next page

Table 3.4 (continued): Summary of related work on unknown attack detection methods.

Reference	Year	ML / DL	Category	Dataset	ID	Payload	Attack Types	Algorithm	Model Size/ Trainable Parameters
[148]	2023	DL	Unsupervised/ Supervised	SynCAN [146], ROAD [119]	✓	✓	13 different attacks	GRU, Latent AE	94MB
[149]	2023	DL	Unsupervised	Survival Analysis Dataset [122], Car Hacking: Attack & Defence Challenge 2020 [2]	✓	✓	Spoofing, Replay, Fuzzy	LSTM-AEs	3.88 - 3.98 MB
[150]	2024	DL	Unsupervised	Survival Analysis Dataset [122]	✓	✓	Flooding, Fuzzy, Malfunction	Transformer	N/A
[151]	2019	DL	Unsupervised	OTIDS [61]	✓	✓	DoS, Fuzzy, Impersonation	ConvLSTM	N/A

Limitations of Unknown Attacks Detection

All proposed anomaly detection IDSs are trained on normal data and use binary classification to classify traffic data as either normal or anomalous by detecting deviations from normal behaviour. In recent years, the frequency of unknown attacks has increased significantly [20], with each new connectivity service introducing additional attack vectors. Moreover, as attackers continuously adapt their methods to bypass existing security measures [21], detecting unknown attacks becomes increasingly challenging, exposing a clear research gap in the literature regarding the robustness of current anomaly based IDSs under dynamic and evolving attack conditions. While detecting previously unseen attacks is crucial, as they may not conform to existing patterns, it is equally important to assign fine grained labels to known attacks. Identifying the specific attack type is highly beneficial for selecting appropriate countermeasures and conducting post attack analysis [28]. Nevertheless, existing approaches do not support such joint capabilities, highlighting an additional research gap in the lack of integrated IDSs that can both detect unknown attacks and classify known attacks while meeting in-vehicle deployment requirements. To address these gaps, the next section discusses work proposed with the ability to detect and classify known attacks while also identifying new, unknown attacks.

3.4.3 Known and Unknown Attacks Detection

To address the limitations of previous approaches and further improve the robustness and detection capability of in-vehicle IDSs, 10 papers found from the search strategy in Section 3.3 developed IDSs capable of identifying both known and unknown attacks [155, 63, 1, 156–162], demonstrating significant advancements in this critical area of cybersecurity. This section reviews state-of-the-art studies and their limitations. Figure 3.7 illustrates exciting work on detecting both known and unknown attacks.

ID-Based Detection

This section reviews research where authors used only CAN IDs as the input feature to develop IDSs for detecting both known and new, unknown attacks.

Hoang et al. [63] and Seo et al. [1] 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 [27].

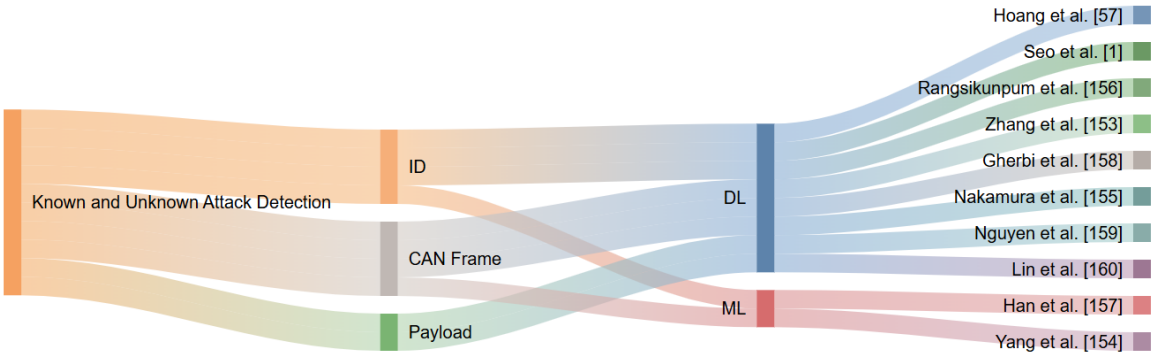


Fig. 3.7 Related work on known and unknown attack detection.

Hoang et al. [63] 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 (GANs). Their IDS was trained on unlabelled data to learn the patterns of normal and malicious data. Only a few labelled samples were used during the subsequent supervised training phase. Even though they use only the CAN ID as the input feature, the number of trainable parameters is 2.15 million for the two models.

Seo et al. [1] have developed a GAN-based IDS (GIDS) for in-vehicle network security. The proposed IDS was trained solely on patterns of CAN IDs extracted from CAN data, which were then converted into simple images. 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.

Rangsikunpum et al. [158] proposed a Binarized Neural Network (BNN)-based IDS (BIDS) for unknown attack detection and known attack classification, utilizing a BNN and a GAN. The model is hierarchically structured into two stages: attack detection in the first stage and known attack classification in the second. To capture sequential patterns in CAN IDs, consecutive CAN IDs are encoded into one-hot vectors and arranged into a 48×48 2D grid. The proposed model is resource-efficient, requiring minimal computational power, making it highly suitable for deployment on low-cost FPGA platforms.

Han et al. [159] proposed an IDS for detecting and identifying abnormalities based on the periodic event-triggered intervals of CAN messages. Statistical features of the event-triggered intervals for each CAN ID, such as mean, variance, quartile deviation, skewness, and kurtosis,

were calculated. These features were then used to train ML models, including DT, RF, and XGBoost to classify attack types. This framework emphasises the event-triggered characteristics of CAN IDs and the statistical moments associated with intervals within a defined time window.

Although using the CAN ID as the only feature reduces the input features and makes the model lightweight, it limits the detection capability of payload manipulation attacks.

Payload-Based Detection

This section discusses IDSs that use the 8-byte CAN payload as an input feature to identify both known and new, unknown attacks.

Zhang et al. [155] 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, the main limitation of the proposed IDS is that it requires either access to the DBC file or knowledge of the CAN payload, which is confidential and proprietary to the vehicle manufacturer [21].

Gherbi et al. [160] proposed a multivariate time series representation matrix to structure CAN data by integrating flow and payload information. They utilised autoencoder-based DL models such as Fully-Connected Networks (FCNs), CNNs, LSTMs, and Temporal Convolutional Networks (TCNs) to extract hierarchical representation vectors from the CAN matrix for anomaly detection. These vectors are derived either from the bottleneck layer in unsupervised tasks or the final layer in supervised tasks. The findings indicate that TCNs and LSTMs achieve strong performance, demonstrating their ability to effectively capture information from the representation matrix during training.

CAN Frame-Based Detection

This section reviews IDSs that use the CAN frame (CAN IDs and payload) as input features to detect both known and new, unknown attacks.

Nguyen et al. [161] propose a semi-supervised learning-based IDS that combines a Variational Autoencoder (VAE) with Adversarial Environment Reinforcement Learning (AERL) for multiclass classification. The proposed IDS is able to detect both known and

unknown attacks. The objective of this approach is to improve training efficiency by reducing the amount of labelled data required.

Nakamura et al. [157] proposed a hybrid model combining a LightGBM-based supervised model and an autoencoder-based unsupervised model to address the challenge of transferring knowledge across multiple car models for detecting and classifying attacks. Time differences between consecutive CAN IDs, along with CAN ID and payload values, were used as input features. Experimental results showed that the hybrid model outperformed the pre-trained LightGBM model.

Lin et al. [162] proposed a two-stage IDS that combines Incremental Learning (IL) and a DNN, referred to as IL-DNN, to address changes in driving environments and behaviours. In the offline training stage, the DNN was applied to actual CAN data to develop a basic classification model. These predicted class labels were then used in the second stage. In the online detection and updating stage, the DNN model was updated using the IL approach with new, unlabeled data, while simultaneously performing intrusion detection. However, this approach risks degrading model performance if the original model's predictions are incorrect. However, both proposed IDSs in [157] and [162] are limited to binary classification, do not consider multi-class classification for known attacks, and do not account for the model size.

Most of the aforementioned studies employ either supervised learning-based methods or unsupervised learning-based methods. To leverage the strengths of both approaches, Yang et al. [156] have developed a multi-tiered IDS, MTH-IDS, to protect intra-vehicle and external networks from cyberattacks. MTH-IDS uses ML algorithms and combines supervised and unsupervised models. The proposed MTH-IDS includes two traditional ML stages: data pre-processing and feature engineering. In the first tier, four tree-based supervised models, DT, RF, ET, and XGBoost, are used to detect known attacks. The second tier incorporates a stacking ensemble model alongside Bayesian Optimization (BO) using the Tree Parzen Estimator (TPE) to enhance the accuracy of the base learners. For unknown attack detection, the third tier introduces a novel unsupervised CL-k-means model. Lastly, the fourth tier applies the (BO) with a Gaussian Process (GP) and two biased classifiers to refine the performance of the unsupervised learners. Despite achieving good results and a small model size of 2.61 MB, the proposed IDS has certain limitations. In the unsupervised model, the authors add an additional tier with two biased classifiers to improve the results. However, training these biased classifiers on False Positives (FPs) and False Negatives (FNs) may lead to poorer performance when testing the model on new, unseen data. Furthermore, adding this tier shifts the model from being purely unsupervised, creating a dependency on labelled datasets, which are often challenging to implement in practical scenarios. Moreover,

the authors used only four features—CAN ID, and selected three features from the payload field, which are DATA[5], DATA[3], and DATA[1], to train the model after feature extraction. However, given that CAN bus data consists of various CAN IDs, each with its own associated payload representing distinct functionalities within the vehicle, this raises the question of whether all CAN IDs share the same important payload features. This consideration leads to our first research question:

RQ1: *Do all CAN IDs share the same important payload features, or should all CAN bus features be considered to ensure system reliability?*

If different CAN IDs have distinct important features, selecting features across all CAN IDs when designing an IDS could risk removing important features for some CAN IDs. This leads to our second research question:

RQ2: *Do feature selection techniques identify important features in CAN bus data for IDS design?*

Together, these questions aim to determine whether selecting specific payload features or considering all payload features is more effective for building a robust IDS. It will also help researchers better understand CAN bus data, enabling them to design IDS models that focus on the most significant features, potentially improving generalization. Although feature selection approaches may lead to more efficient models, they create the risk that attackers could manipulate features not considered during the model's training process [163]. This presents a critical limitation in CAN bus data for three reasons. First, selecting a subset of CAN bus payload features as important while discarding others could allow attackers to exploit the neglected features and bypass the model [164, 165]. Second, the evolving landscape of attack scenarios means that features chosen to detect one category of attack may become outdated or insufficient to address new, unseen attacks [163].

Yang et al. [156] employed conventional ML models in their proposed IDS due to their lower computational cost compared to DL algorithms. However, DL has shown superior performance in processing large volumes of data efficiently and at a faster rate [42]. Considering that modern vehicle ECUs produce around 2,000 CAN frames per second [1], this capability is essential to handle the extensive data of the CAN bus. Moreover, multiple studies have found that DL-based IDSs outperform traditional ML-based IDSs in automotive applications [166]. 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 [155]. Additionally, traditional ML often requires manual feature engineering, such as applying correlation-based feature selection, which can be time-consuming [69]. In contrast, DL automatically deduces features, allowing algorithms to directly discern optimal features from raw data [167]. Furthermore, DL-based IDSs are especially capable of detecting new unknown attacks and can scale more effectively to highly complex in-vehicle network data while maintaining efficacy [167]. This leads to our third and fourth research questions:

RQ3: *How can an in-vehicle IDS be designed to detect both known and unknown attacks using DL algorithms?*

RQ4: *How can the IDS remain lightweight and meet real-time requirements while operating within resource constraints?*

Table 3.5 summarises the details of known and unknown attack detection studies.

Limitations of Existing known and Unknown Attacks Detection

Although we have discussed the limitations of each previous work in the previous section, a common limitation of all the proposed approaches pertains to their deployment strategy. All these work have deployed their IDSs using a traditional centralised learning approach, which involves transmitting large volumes of data to the cloud for both training and testing on the CAN bus. This method raises significant issues, such as privacy concerns, high communication overhead, and longer response times [29]. In the absence of explicit privacy-preserving mechanisms, sensitive vehicle and driver data may be exposed, increasing the risk of privacy breaches and regulatory non-compliance. In addition, continuous data transmission introduces significant communication overhead and dependence on cloud connectivity, which can delay detection responses. Despite the high-frequency nature of CAN traffic, communication cost requirements are rarely reported. Moreover, although some studies measure IDS processing latency, they do not evaluate the additional response delay introduced by cloud communication. Consequently, while centralised IDSs may achieve high detection accuracy, their deployment in real in-vehicle environments remains challenging due to privacy, latency, and communication constraints.

Table 3.5 Summary of related work on known and unknown attack detection methods.

Reference	Year	ML / DL	Category	Dataset	Algorithm	M-C	ID	Payload	Model Size/ Trainable Parameters
ID-Based Attack Detection									
[63]	2022	DL	Semi-supervised	Car-Hacking [1]	AE, GAN		✓		2.15 million
[1]	2018	DL	Unsupervised	Car-Hacking [1]	GAN		✓		N/A
[158]	2024	DL	Semi-supervised	Car Hacking [1], Survival Analysis Dataset [122]	BNN, GAN	✓	✓		4.07 Mb
[159]	2021	ML	Unsupervised	Own	DT, RF, XGBoost	✓	✓		N/A
Payload-Based Attack Detection									
[155]	2019	DL	Supervised	Simulation	DNN			✓	N/A
[160]	2020	DL	Supervised/ Unsupervised	SynCAN [146]	FCN, CNN, TCN, LSTM, AE			✓	0.01 - 0.3MB
CAN Frame-Based Attack Detection									
[157]	2021	DL	Unsupervised	Survival Analysis Dataset [122]	LightGBM, AE		✓	✓	N/A
[161]	2024	DL	Semi-supervised	car-hacking [85], ROAD [119]	VAE and AERL	✓	✓	✓	2,542 KB
[162]	2021	DL	Supervised/ Semi-supervised	car-hacking [85]	DNN and IL		✓	✓	N/A
[156]	2022	ML	Hybrid	Car-Hacking [1], CICIDS2017 [168]	DT, RF, ET, XGBoost, CL-k-means	✓	✓	✓	2.61 MB

M-C: Multi-class classification.

3.4.4 Evaluation Metrics

In this section, we review all the evaluation metrics used to assess the proposed models in previously reviewed papers. The aim is to emphasise the importance of considering these metrics when designing models, rather than focusing on a few while ignoring others, to develop more deployable solutions. Based on the reviewed papers, we categorise the evaluation metrics into performance metrics, time complexity metrics, memory requirement metrics, and other metrics.

Performance metrics assess a model's effectiveness, including accuracy, F1-score, precision, recall (also known as Detection Rate (DR)), Error Rate (ER), confusion matrix, False Negative Rate (FNR), True Positive Rate (TPR), False Positive Rate (FPR), and True Negative Rate (TNR), also known as specificity. Additionally, False Alarm Rate (FAR), Receiver Operating Characteristic (ROC) Curve, Area Under the ROC Curve (AUC-ROC), and Area Under the Precision-Recall Curve (AUPR) are commonly used. These metrics are computed using True Positives (TP), False Positives (FP), False Negatives (FN), and True Negatives (TN). Furthermore, the G-mean score and Matthews Correlation Coefficient (MCC) are valuable for evaluating model performance, particularly in cases of significant class imbalance [131, 33]. Other relevant metrics include kappa and loss. For time complexity, several measures are commonly used, including training time, detection (inference) time, and latency. Regarding memory requirement metrics for evaluating model size, key metrics include the number of trainable parameters (which reflects memory usage), the model size in megabytes or kilobytes, and the number of Floating Point Operations (FLOPs). Other metrics, which are less commonly used in the reviewed papers, include resource allocation, power consumption, and Multiply-Accumulate (MAC) operations, which measure the speed of DL models [94].

Tables 3.6 and 3.7 present the evaluation metrics used in the reviewed papers. Most studies have focused on some performance metrics while giving less consideration to time and memory requirements. Considering all these metrics (performance, time, and memory) makes the proposed models more deployable and easier to compare with other works.

Table 3.6 Performance metrics used in existing works.

Metric	Work(s)
Performance Metrics	
Accuracy	[104, 95, 30, 9, 156–159, 32, 161, 147, 145, 118, 149, 65, 86–89, 103, 92], [93, 66, 9, 31, 98, 143, 1, 101, 106–111, 113–115, 162, 135–141, 132, 100, 33]
F1-score	[98–118, 132, 142–145, 151, 135, 156–162, 63, 133, 147–150], [30, 85, 86, 89, 91, 93, 66, 65, 9, 31, 130, 88, 94, 95, 129, 9, 137–141]
Precision	[157, 161, 160, 145, 151, 147, 158, 1, 92, 142, 30, 143, 130, 149, 9, 150, 63], [102–118, 98, 99, 85, 144, 86, 88, 89, 91, 93–95, 66, 65, 133–141]
Recall	[157, 160, 161, 151, 147, 158, 150, 130, 86, 108–118, 63, 103–107, 99, 98], [85, 88, 89, 91, 66, 9, 95, 94, 102, 100, 30, 149, 133, 135–145, 65, 156, 1]
Confusion Matrix	[128, 130, 105, 96, 85, 109, 88, 66, 93, 94, 103, 98, 115–118, 111], [113, 155–158]
TPR	[66, 92, 31, 145, 9, 146, 95, 101, 148, 155]
FPR	[93, 32, 129, 161, 31, 9, 65, 95, 89–91, 66, 115, 100, 101, 145, 130, 143], [136, 105, 148, 155]
FNR	[65, 130, 9, 143, 85, 93, 89, 161, 90, 115, 99, 33, 94, 148, 129, 104, 136]
TNR	[93, 146, 9, 148, 117]
ER	[85, 104, 63, 133, 118]
ROC	[90, 132, 128, 151, 150, 106, 103, 137, 118, 91, 96, 155]
AUC-ROC	[9, 134, 105, 149, 86, 66, 93, 104, 138, 107, 115, 95, 97, 33, 114, 146], [147, 159, 117, 1, 141, 144]
AUPR	[134]
MCC	[33, 145]
TP, FP	[87, 128]
FN, TN	[128]
G-mean Score	[131]
FAR	[156]
Kappa	[138]
Loss	[106, 108]

Table 3.7 Other evaluation metrics used in existing works.

Metric	Work(s)
Time Complexity Metrics	
Training Time	[85, 98, 108, 138, 114, 118, 128, 30, 155, 160, 90, 156]
Detection Latency/ Inference Time	[85, 88, 90, 94, 148, 31, 98, 106, 118, 128, 145, 129, 130, 138, 132, 133], [33, 136, 30, 142, 32, 143, 149, 150, 63, 1, 158, 155, 161, 112, 162, 156]
Execution Time	[92, 96, 100, 101, 111, 113, 159, 155]
Memory Requirements Metrics	
Memory Footprint (Size)	[88, 31, 133, 33, 30, 32, 144, 148, 149, 158, 160, 138, 161, 156]
Parameters	[94, 33, 30, 148, 63, 160, 8]
Training Cost (FLOPS)	[118, 30]
Other Metrics	
Resource Utilization	[143, 32, 130, 88]
Power/Energy Consumption	[30, 130, 88]
MAC	[94]

3.5 Related Work on FL-Based IDSs for In-Vehicle Networks

This section reviews existing FL-based in-vehicle IDSs and highlights their limitations. Table 3.8 summarises the eight papers collected from the search strategy in Section 3.3.

Table 3.8 FL-based IDSs for in-vehicle network.

Reference	Aggregation Function	Dataset	FL Implementation
[169]	FedAvg	car-hacking [85]	PyTorch
[37]	FedAvg	HCRL CAN Intrusion Detection [61]	N/A
[36]	FedAvg	car-hacking [85], NAIST CAN attack dataset[66]	Keras, TensorFlow
[38]	FedAvg, FedProx	READ [170]	N/A
[35]	FedAvg	Car Hacking: Attack & Defence Challenge 2020 [2]	Keras, TensorFlow
[39]	FedAvg	HCRL CAN Intrusion Detection [61]	N/A
[145]	FedAvg, FedProx	Recan [154]	N/A
[171]	FedAvg	Own	Pytorch, Flower

Driss et al. [35] introduced an FL-based framework to identify attacks in vehicular sensor networks. Given the resource constraints of smart sensing devices in vehicular networks, the authors emphasised the significance of adopting lightweight security approaches. To address this issue, they utilised a set of Gated Recurrent Units (GRU) with an ensemble method based on RF to aggregate the global ML models. They distributed the dataset equally among clients.

Shibly et al. [36] proposed a personalised FL-based IDS that eliminates the need for data sharing. The authors explored both supervised and unsupervised methods within the FL framework, including CNN, XGBoost, MLP, and AE. Although their results were promising for both binary and multiclass classification, they did not account for Non-Independent and Identically Distributed (non-IID) data distributions.

Yu et al. [37] introduced an FL-based IDS based on LSTM for in-vehicle networks. They utilised the periodicity of CAN communications to predict the arbitration IDs of new messages. The 11-bit arbitration ID is transformed into vectors using one-hot encoding, and these vectors are utilised by the LSTM to predict the subsequent arbitration ID. Data is evenly distributed among clients, with each client holding 1,000 instances for training and 200 instances for testing. A comparison between the FL-based and a centralised IDS revealed a 0.071 accuracy loss for the FL-based IDS. However, the authors suggested that this loss could be mitigated through a cumulative error scheme.

Zhang et al. [38] developed an anomaly detection system utilising a graph neural network, capable of detecting CAN bus intrusions within a minimal 3-millisecond timeframe. They

construct a two-stage classifier cascade consisting of a classifier designed for anomaly detection in one class, and another classifier for categorising attacks into multiple classes. To address novel anomalies from unseen classes, an openmax layer is integrated into the multi-class classifier.

Yang et al. [39] introduced an IDS for in-vehicle networks utilising federated DL. The proposed approach leverages the periodicity of network messages, employs the ConvLSTM model, and trains the model through federated DL. To simulate a non-IID environment, clients were assigned varying data samples (ranging from 50 to 3500), although specific details regarding data distribution among clients and the distribution of samples across all classes were not provided.

Taslimasa et al. [169] proposed ImageFed, a practical IDS designed to preserve privacy, which utilises federated CNNs. To simulate a non-IID setting, data were distributed to vehicles using a Dirichlet(μ) distribution, with (μ) values varying from 0.1 to 0.7. To assess ImageFed's resilience, they examined two potential scenarios leading to performance decline in FL: non-IID clients and limited availability of training data. Table 3.8 summarises previous work and highlights our contribution. In cases where the aggregation function is not explicitly indicated, as in [35, 36], we assume that FedAvg is the method employed.

Longari et al. [145] deployed their proposed IDS, CANdito, presented in Section 3.4.2, in an FL setting to evaluate its detection efficiency and communication overhead, comparing it to a centralised version of the same algorithm. Experimental results suggest that FL could be a suitable approach in real-world scenarios where data privacy and security cannot be ignored. While the detection capabilities of the federated model are slightly lower than those of the centralised model, it still demonstrates robust performance.

To overcome the challenge of DL models requiring large amounts of data to achieve optimal performance—particularly in the case of CAN bus IDS—Hoang et al. [171] proposed CANPerFL, an IDS that employs a personalised FL approach to aggregate datasets from different car models. Their approach builds a universal model trained on a small amount of data from each manufacturer, providing global knowledge that enhances the performance of individual participants. Experimental results show that the proposed model improves F1 scores by 4% overall compared to baselines. Moreover, it offers significant advantages when the local dataset of each participant is relatively small.

3.5.1 Limitations of Existing FL-Based IDSs

Although previous studies [169, 35–39] have deployed in-vehicle IDSs in FL environments, they face several limitations. All existing works rely on a standard cloud-based FL architecture, coordinated by a single central aggregator for training and evaluation. However, this reliance restricts generalisation, as the central server often struggles to capture diverse driving behaviours and adapt to new attack patterns beyond its coverage. For instance, attacks detected by server A within its region remain unknown to server B, and similar challenges arise when adapting to heterogeneous driving behaviours. This limitation gives rise to our fifth research question:

RQ5: *How can in-vehicle IDSs detect new attacks that were never part of the central server's coverage in the FL architecture?*

Beyond limited generalisation, reliance on a single central aggregator also raises practical challenges. Model sharing between the server and millions of vehicles causes network congestion and communication delays [40], which slow down the learning process and delay timely updates for intrusion detection. This leads to our sixth research question:

RQ6: *How can the architecture of FL-based in-vehicle IDSs be enhanced to mitigate network congestion and communication delays?*

Finally, dependence on a single aggregator creates a critical vulnerability, as failure of the central server compromises the entire system. This motivates our seventh research question:

RQ7: *How can we reduce the risk of single-point failure inherent in centralised FL architectures?*

Together, these research questions aim to advance FL-based in-vehicle IDSs towards architectures that generalise across diverse behaviours, detect new attacks beyond the central server's coverage, maintain efficient communication, and remain resilient to critical failures.

3.6 Conclusion

The main contribution of this chapter was a comprehensive survey of existing ML and DL approaches for building in-vehicle IDSs, focusing on detecting known, unknown, and combined known and unknown attacks. Additionally, we reviewed research on FL-based

IDSs applied to in-vehicle networks. The discussion of prior work identifies research gaps and formulates key research questions, which are outlined in the subsequent sections.

RQ1 and RQ2: Chapter 4 contributes to addressing RQ1 and RQ2 by analysing CAN bus data, comparing the results of various feature selection methods on different datasets, and gaining critical insights into whether feature selection should be used when building a robust in-vehicle IDS.

RQ3 and RQ4: Chapter 5 also contributes to addressing RQ3 and RQ4 by developing a lightweight and novel multi-stage IDS for in-vehicle network security, capable of detecting both known and unknown attacks, and addressing limitations in previous work.

RQ5, RQ6 and RQ7: Chapter 6 contributes to addressing RQ5, RQ6 and RQ7 by deploying our proposed in-vehicle IDS in a hierarchical FL framework to enhance the robustness of the IDS.

As discussed in this chapter, different studies use different CAN bus features to build their in-vehicle IDS: some rely only on the CAN ID, while others use payload data. Some approaches utilise the entire CAN frame, which includes both the CAN ID and the payload, while other studies apply feature selection methods to identify the most relevant features and build a lightweight IDS. However, given the low dimensionality of CAN bus data, the next chapter explores the reliability of feature selection for in-vehicle IDS, laying the groundwork for designing a more robust system.

Chapter 4

Evaluating the Reliability of Feature Selection for CAN Bus IDSs [1]

Feature selection techniques are commonly employed to enhance IDS performance. However, in the context of CAN bus data, where each CAN ID has unique payload values and patterns, this approach may be ineffective. This chapter answers the following questions: Do all CAN IDs share the same important payload features, or should all CAN bus features be considered to ensure system reliability? If so, do feature selection techniques identify important features in CAN bus data for IDS design? Experimental results reveal that many CAN IDs contain zero-constant values that remain unchanged throughout the dataset. Furthermore, applying various feature selection techniques often identifies these constant zeros as important features. This finding suggests that selecting a consistent set of important features across all CAN IDs is impractical, as uniform feature selection may inadvertently exclude critical information. This chapter demonstrated that retaining all CAN payload features yields more stable and robust intrusion detection performance than using a reduced feature set. Comparative experiments showed that the full feature configuration achieves consistently higher and less variable performance across repeated train–test splits, supporting its suitability for robust in-vehicle IDS design.

4.1 Introduction

Feature selection refers to the process of obtaining a subset of the original feature set based on defined selection criteria. It serves as a critical pre-processing step in ML, aiming to reduce computational complexity by eliminating irrelevant features while maintaining or

[1] This chapter is published in *International Conference on Cyber Security, Privacy in Communication Networks*. Singapore: Springer Nature Singapore, 2023.

improving IDS performance [172]. This approach offers numerous advantages, including reducing overfitting, facilitating data visualisation, decreasing model training time, and minimising storage requirements [173, 174]. Effective feature selection can also enhance learning accuracy [175]. However, as the overarching aim of this thesis is to design a robust and lightweight in-vehicle IDS, an important decision is determining which features to select. Should all features be considered, or is it sufficient to rely on a subset of selected features? Given that different CAN IDs have varying payload values, is it expected that all CAN IDs share the same important features? This chapter aims to address these questions by analysing CAN bus data and evaluating the effectiveness of various feature selection techniques in this context. Additionally, it seeks to enhance researchers' understanding of CAN bus data to support the development of more robust IDS solutions. To our knowledge, no prior work has specifically explored feature selection for CAN bus data. This chapter aims to answer the following research questions.

RQ1: *Do all CAN IDs share the same important payload features, or should all CAN bus features be considered to ensure system reliability?*

RQ2: *Do feature selection techniques identify important features in CAN bus data for IDS design?*

In answering these questions, the following contributions are made:

C1 To provide insight into CAN bus data by analysing it to support the design of a robust IDS for in-vehicle networks.

C2 To evaluate the feasibility and effectiveness of feature selection techniques for building a robust IDS for in-vehicle networks.

4.2 Methodology

In this section, we present the experimental setup used in our experiments. We then analyse the CAN bus data to gain insights, introduce the feature selection techniques applied, and describe the datasets used for comparison and evaluation.

4.2.1 Experimental Setup

The implementation was carried out in Google Colab, a web-based editor developed by Google Research that allows users to write and execute Python code directly from the browser. The experimental setup included a 64-bit Ubuntu 20.04.5 LTS operating system, an 11th Gen Intel(R) Core(TM) i7-11700 @ 2.50GHz processor, 31.1 GiB of RAM, an NVIDIA GPU, and Python version 3.9.16.

4.2.2 CAN Bus Data Analysis

In this section, we analysed the CAN bus data to gain insights into important features. Through close examination, we observed that each CAN ID exhibits distinct patterns in data field values (payload) (D0, D1, D2, D3, D4, D5, D6, D7), with features appearing as zero-constant values, constant non-zero values, or changing values. For example, as shown in Table 4.1, CAN ID 399 consistently has zeros in data fields D2, D3, D4, D6, and D7, which remain constant throughout the dataset. Table 4.1 highlights these zero-constant values in blue, representing data that do not change. In contrast, values highlighted in yellow indicate variability, while those in green signify constant non-zero values.

Table 4.1 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

4.2.3 Feature Selection Techniques

We selected six widely used feature selection methods, including univariate selection, extra trees classifier, information gain, Recursive Feature Elimination (RFE), chi-square test, and Pearson correlation.

1. **Univariate selection:** The univariate selection method applies statistical tests to identify features that have the strongest relationship with the target variable, facilitating

the elimination of less relevant features. The SelectKBest library [176] is used to select the top K features that are most relevant to the target variable, where K can be specified based on the requirements.

2. **Extra trees classifier:** The extra trees classifier is an ensemble learning method that combines the outputs of multiple de-correlated decision trees, collectively forming a "forest" to generate the final classification result. It constructs numerous decision trees using the entire training dataset. At each root node, the extra trees algorithm randomly selects a subset of features k and determines a partially random cut point to define the split rule. This rule splits the parent node into two child nodes, and the process repeats recursively for each child node until a leaf node is reached, where no further splits occur. The final prediction is determined by aggregating the outputs of all trees through a majority vote. For feature selection, features are ranked in descending order based on their Gini importance, allowing users to select the top k features according to their specific needs as inputs for the classification model [177].

3. **Information gain :** Information gain is an information-theoretic metric that quantifies the amount of information a feature contributes to predicting the target class, expressed as the number of bits of information gained. Feature selection using information gain is a widely used approach due to its simplicity and ease of interpretation. This method evaluates the dependency between each feature and the target label, assigning higher information gain values (up to 1) to features that are more informative or relevant for predicting the target variable. However, as information gain evaluates each feature independently, it cannot address feature redundancy [178].

4. **Recursive feature elimination (RFE):** RFE is a feature selection technique that iteratively removes the weakest feature(s) until the desired number of features is reached. It employs a backward elimination strategy based on feature ranking. Initially, RFE fits a model using all available features, calculates their scores, and ranks them accordingly. The feature with the lowest rank is then removed, and the process is repeated with the remaining features. The procedure continues until an optimal subset of features is identified, with the subset size serving as a tunable parameter [178].

5. **Chi-square test:** Chi-square test is based on statistical measures that evaluate features independently in relation to the target class. The Chi-square score is useful for assessing independence and goodness of fit. A higher Chi-square value indicates a feature that is more relevant to the class label [178].

6. **Pearson correlation:** Pearson correlation is a univariate correlation criterion that computes test statistics to measure the statistical relationship, or association, between two continuous variables [178]. This measure provides insight into both the magnitude of the association (correlation) and the direction of the relationship. There are two types of correlation: positive correlation, where an increase in one variable corresponds to an increase in the other (or a decrease in one corresponds to a decrease in the other), indicating a linear relationship where the variables move in tandem; and negative correlation, where an increase in one variable corresponds to a decrease in the other, and vice versa. These correlations help identify the strength and nature of the linear relationship between features.

This was done to assess whether these methods would identify the zero-constant features observed in Section 4.2.2 within each CAN ID's data as important features. These methods were chosen because they are among the most widely used in the field. We apply each feature selection method to individual CAN IDs and compare the results with the zero-constant and non-constant features.

4.2.4 Datasets

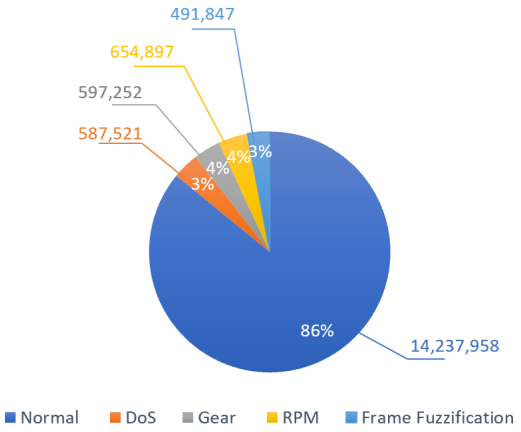
To ensure the validity of our observations across various CAN bus datasets, we validate them using three widely recognised benchmark datasets in automotive security research: the Car Hacking Dataset [1], the Car Hacking: Attack & Defence Challenge 2020 Dataset [2], and the CAN Dataset for Intrusion Detection (OTIDS) [61]. All datasets used in this study were obtained from the Hacking and Countermeasure Research Lab (HCRL), which focuses on data-driven security research. A key strength of HCRL is the availability of high-quality datasets collected from real-world vehicular environments, making them particularly suitable for realistic intrusion detection evaluation. These datasets encompass both normal and attack data, providing detailed information for each CAN message, including timestamp, CAN ID, DLC, data field, and flag. The timestamp indicates the precise recording time of each message from start-up, while the CAN ID determines message priority, with lower values taking precedence over higher ones. Additionally, the DLC specifies the data field length in bytes (up to 8 bytes), and the flag distinguishes between normal and attack messages. Table 4.2 summarises the CAN bus message features, data types, and value ranges, which are consistent across all these datasets. In this study, we focus on using the CAN ID to categorise data and the data fields, which represent the payload values. Figure 4.1 shows that all three datasets exhibit class imbalance between normal and attack instances. The

Table 4.2 CAN bus message features, data types, and value ranges.

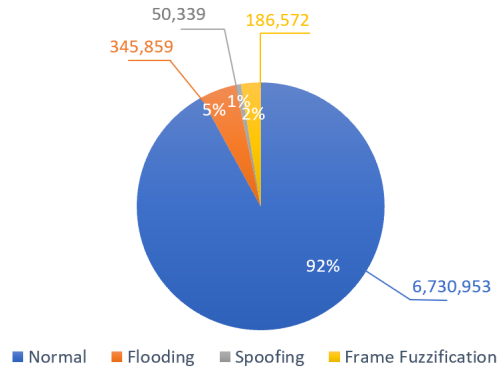
Feature	Type	Range / Description
CAN ID	hexadecimal	Dataset dependent
Timestamp	floating point	Continuous time value
DLC	integer	0–8
DATA[0–7]	integer (byte)	0–255
Flag	categorical	Normal or attack class

most severe imbalance is observed in the Car Hacking: Attack and Defence Challenge 2020 dataset [2], where normal traffic accounts for 92% of the data and all attack classes together comprise only 8%, as illustrated in Figure 4.1b. This dataset contains a total of 7,313,723 instances. A lower degree of imbalance is present in the Car Hacking dataset [1], where attack instances represent 14% of the total data. This dataset is substantially larger, containing 14,237,958 instances. The least imbalance is observed in the CAN Dataset for Intrusion Detection (OTIDS) [61], in which attack traffic constitutes 34% of all samples. This dataset also has the smallest size, with a total of 3,618,437 instances. These imbalance characteristics reflect realistic in-vehicle conditions, where malicious events are rare compared to benign traffic. These statistics highlight that CAN IDS evaluation must cope with severe class imbalance, motivating the use of imbalance robust evaluation metrics or balancing strategies. Consequently, accuracy alone is insufficient for evaluation, and metrics such as the F1-score are more appropriate for assessing detection performance under skewed class distributions.

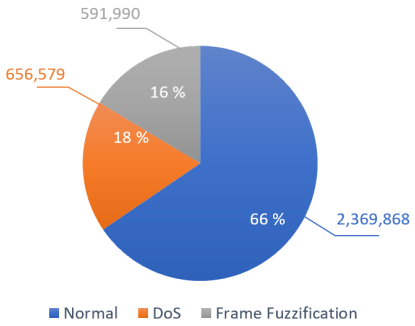
For a more detailed examination of the datasets, Table 4.3 summarises the distribution of normal and attack CAN IDs across the evaluated datasets. Across all datasets, attack traffic is generated by a substantially smaller number of CAN IDs compared to normal traffic. In particular, DoS and flooding attacks rely on a single CAN ID in all cases. Moreover, spoofing attacks, such as gear and RPM spoofing, involve a limited and fixed set of CAN IDs, reflecting realistic adversarial behaviour that closely mimics legitimate messages. In contrast, frame fuzzification affects a large number of CAN IDs, most notably in the Car Hacking Dataset, where 2048 IDs are involved, highlighting its indiscriminate and noisy nature. The Car Hacking Dataset and the Attack and Defence Challenge 2020 dataset contain multiple attack categories with distinct CAN ID footprints, making them suitable for evaluating multi-class IDSs. OTIDS, by contrast, includes fewer attack types and CAN IDs, suggesting a more controlled but less diverse threat model.



(a) Car Hacking Dataset [1]



(b) Car Hacking: Attack & Defence Challenge 2020 Dataset [2]



(c) Dataset for Intrusion Detection (OTIDS) [61]

Fig. 4.1 Class distribution across the evaluated CAN bus datasets.

Table 4.3 Distribution of normal and attack CAN IDs across datasets.

Dataset	Category	Number of CAN IDs
Car Hacking Dataset [1]	Normal	38
	DoS	1
	Gear spoofing	26
	RPM spoofing	26
	Frame Fuzzification	2048
Car Hacking: Attack & Defence Challenge 2020 Dataset [2]	Normal	68
	Flooding	1
	Spoofing	4
	Frame Fuzzification	80
CAN Dataset for Intrusion Detection (OTIDS) [61]	Normal	45
	DoS	1
	Frame Fuzzification	45

4.3 Results and Analysis

As shown in Table 4.4, various CAN IDs have different zero-constant features, ranging from none, as seen in CAN ID 608, to all features being constant zeros, as observed in CAN ID 1072. This observation has been consistently validated across all three datasets. This observation suggests that significant features (i.e., those with variability) for one CAN ID may not be relevant for another, making it impractical to select a consistent subset of important features across all CAN IDs due to the inherent nature of CAN bus data. Table 4.5 presents the important features identified by each feature selection method, compared to the zero-constant values and non-constant values. We applied these methods to a representative subset of five CAN IDs (2, 120, 399, 1201, and 1442) as sample data. For example, for CAN ID:2, the (non-constant) values are D[5], D[6], and D[7]. To determine whether constant features have been selected by other feature selection methods, we choose an equal number of features. For example, if the (non-constant) features are four features, we also select the four top features of the other methods. In univariate selection, Pearson correlation, and chi-square test methods, the important selected characteristics are D[2], D[3], and D[4], all of which are constant. In the extra trees classifier method, the crucial features include D[7], D[3], and D[6], with D[3] being constant. Furthermore, the information gain method highlights significant features, such as D[3], D[0], and D[1], while the RFE method identifies important features as D[2], D[6], and D[7]. Based on the results, all comparable feature selection methods choose some or all of their important features, encompassing zero-constant features designed to handle irrelevant information.

Table 4.4 Samples of the number of constant zeros in each CAN ID.

Car Hacking Dataset [1]		**Car Hacking: Attack & Defence Challenge 2020 [2]**		**OTIDS Dataset** [61]	
CAN ID	**Data Field**	**CAN ID**	**Data Field**	**CAN ID**	**Data Field**
2	D[0], D[1], D[2], D[3], D[4]	67	D[1], D[5], D[6]	24	D[1], D[5]
160	D[4], D[7]	304	D[4]	66	D[1], D[4], D[5], D[6], D[7]
161	D[2], D[3], D[5], D[6], D[7]	320	D[2], D[5]	67	D[1], D[2], D[3], D[4], D[5], D[6], D[7]
261	D[1], D[3]	339	D[4]	68	D[0], D[1], D[2], D[5], D[6], D[7]
305	D[3]	854	D[0], D[1], D[2], D[5], D[6], D[7]	129	D[3], D[4], D[5], D[6]
320	D[1], D[2], D[3]	871	D[2], D[5]	160	D[4], D[6]
339	D[0], D[4], D[6], D[7]	872	D[0], D[3]	161	D[5], D[6], D[7]
399	D[2], D[3], D[4], D[6], D[7]	897	D[3], D[4]	272	D[4], D[5], D[6], D[7]
497	D[1], D[2], D[3], D[4], D[5], D[6], D[7]	903	D[4], D[7]	339	D[0], D[4]
672	D[1], D[7]	909	D[0], D[1], D[3]	356	D[0], D[2], D[3], D[4], D[5]
704	D[1], D[2], D[3], D[4], D[5], D[6], D[7]	913	D[0], D[1], D[2], D[3], D[4], D[5]	357	D[3], D[4], D[5]
790	D[6]	916	D[3]	399	D[0], D[3], D[4], D[6], D[7]
809	D[6]	1040	D[3], D[5], D[6], D[7]	848	D[5], D[6]
848	D[5], D[6]	1042	D[1], D[2], D[5], D[6], D[7]	880	D[2], D[5], D[6]
880	D[0], D[2], D[4], D[5], D[6], D[7]	1056	D[7]	898	D[3], D[4], D[5], D[6]
1072	D[0], D[1], D[2], D[3], D[4], D[5], D[6], D[7]	1057	D[2], D[6]	1087	D[7]
1088	D[1], D[2], D[3], D[7]	1069	D[0], D[1], D[2], D[3]	1088	D[2], D[3], D[7]
1201	D[4], D[5], D[6]	1151	D[4], D[6]	1264	D[0], D[1], D[3], D[4]
1264	D[0], D[4]	1162	D[0], D[1]	1265	D[2], D[3], D[4], D[5], D[6], D[7]
1349	D[2], D[4], D[5], D[6], D[7]	1164	D[3], D[4], D[5], D[6], D[7]	1266	D[0], D[1], D[4], D[5], D[6]
1440	D[0], D[1], D[2], D[3], D[4], D[5], D[6], D[7]	1168	D[0], D[1], D[4]	1306	D[0], D[2], D[3], D[4], D[5], D[6], D[7]
1442	D[1], D[4], D[5], D[6], D[7]	1170	D[3], D[4], D[5], D[6]	1349	D[2]
1680	D[0], D[1], D[3], D[6], D[7]	1173	D[0], D[1]	1415	D[0], D[1], D[2], D[3], D[4], D[5], D[6]
1697	D[0], D[1], D[2], D[3], D[4], D[5]	1183	D[2], D[3], D[4], D[6], D[7]	1435	D[0], D[1], D[2], D[3], D[4], D[5], D[6]
2009	D[4], D[5], D[6], D[7]	1280	D[2], D[3], D[5]	1680	D[1], D[5]
2024	D[4], D[5], D[6], D[7]	1322	D[1], D[2], D[3], D[5]	1201	D[0], D[2], D[3], D[4], D[5], D[6]

Table 4.5 Important features selected by each feature selection method.

Feature Selection Method	**Selected Features**				
	ID: 2	ID: 160	ID: 399	ID: 1201	ID: 1442
Univariate selection	D[3], D[4], D[2]	D[3], D[7], D[1], D[4], D[0], D[2]	D[0], D[3], D[7]	D[3], D[4], D[5], D[6], D[2]	D[2], D[3], D[7]
Extra trees classifier	D[7], D[3], D[6]	D[1], D[3], D[4], D[6], D[0], D[7]	D[0], D[3], D[2],	D[3], D[4], D[7], D[5], D[2]	D[3], D[7], D[4]
Information gain	D[3], D[0], D[1]	D[3], D[1], D[0], D[7], D[4], D[2]	D[0], D[1], D[2],	D[3], D[7], D[0], D[1], D[4]	D[3], D[7], D[0]
Pearson correlation	D[1], D[2],[3]	D[2], D[3], D[4], D[5], D[6], D[7]	D[2], D[3], D[4]	D[1], D[2], D[4], D[5], D[6]	D[2], D[3], D[4]
RFE	D[2], D[6], D[7]	D[1], D[3], D[4], D[5], D[6], D[7]	D[0], D[4], D[7]	D[3], D[4], D[5], D[6], D[7]	D[3], D[6], D[7]
chi-square test	D[2], D[3], D[4]	D[0], D[1], D[2], D[3], D[4], D[7]	D[0], D[3], D[7]	D[2], D[3], D[4], D[5], D[6]	D[2], D[3], D[7]
Zero-constant Values	D[0], D[1], D[2], D[3], D[4]	D[4], D[7]	D[2], D[3], D[4], D[6], D[7]	D[4], D[5], D[6]	D[1], D[4], D[5], D[6], D[7]
Non-Constant Values	D[5], D[6], D[7]	D[0], D[1], D[2], D[3], D[5], D[6]	D[0], D[1], D[5]	D[0], D[1], D[2], D[3], D[7]	D[0], D[2], D[3]

The main finding of this work is that for each CAN ID, there are zero-constant values that never change in the CAN bus data. Compared to all feature selection methods we tested, these zero-constant features are chosen as important and remove genuinely important features instead. Therefore, security solution designers should carefully choose the right feature selection method. Also, choosing a feature selection for all CAN IDs can lead to information loss since some features that are important for one CAN ID would be zero-constant for other CAN IDs. While feature selection is a crucial pre-processing step before applying ML algorithms, in the context of CAN bus data, it may be ineffective due to several reasons. Each CAN ID exhibits unique patterns and important features, making it challenging to apply a universal feature selection method. The presence of zero-constant values across different features adds to the complexity, potentially leading to the exclusion of important features for some CAN IDs while misclassifying zero-constant features as important for others. Furthermore, the use of feature selection on CAN bus data may unintentionally create opportunities for adversaries to inject malicious data into the removed or less prioritised features [164, 165]. In this study, we addressed two key research questions: **RQ1:** Do all CAN IDs share the same important payload features, or should all CAN bus features be considered to ensure system reliability? and **RQ2:** Do feature selection techniques accurately identify important features in CAN bus data for IDS design? Our experiments demonstrated that different CAN IDs emphasise different payload features, and that incorporating all CAN ID features is the more reliable approach to avoid losing important information when designing a robust in-vehicle IDS. The main findings of this chapter are summarised as follows:

- For each CAN ID in CAN bus data, payload values can be zero-constant, constant non-zero, or changing, depending on how the CAN ID is defined in the dictionary.
- When applying various feature selection methods to each CAN ID, most of them identify these zero-constant features as important while neglecting other changing features, resulting in the loss of important features and the selection of features that are always zero.
- Applying feature selection methods to CAN bus data (across all CAN IDs) leads to a loss of important information, as features that are important for one CAN ID may be zero-constant for others.

- This analysis can help researchers gain a deeper understanding of CAN bus data and leverage our observations to design a customised feature selection method specifically tailored for CAN bus data.

4.4 Robustness Analysis of Feature Selection

This section extends the analysis of feature selection through a comparative evaluation against the approach proposed by Yang et al. [156], which represents a key reference in the in-vehicle intrusion detection domain in general and an important point of comparison for this thesis. While Yang et al. advocate the use of a reduced feature set comprising four CAN bus features, namely CAN ID, DATA[5], DATA[3], and DATA[1], this thesis argues that, given the low dimensionality and structured nature of CAN bus data, all payload features contain meaningful information. Consequently, aggressive feature reduction may adversely affect the robustness of the IDS. To investigate this claim, a series of controlled experiments was conducted using a Random Forest (RF) classifier as a simple and widely adopted ML model. RF is employed here as an analytical tool rather than as the final IDS, enabling an interpretable comparison of different feature configurations within a conventional learning framework. The choice of RF is motivated by its extensive use in the in-vehicle intrusion detection literature, as discussed in Chapter 3. The remainder of this section details the experimental design and evaluation methodology, followed by an analysis of performance consistency and variability across repeated train–test partitions. It then examines the stability of feature importance across splits and across datasets, and concludes with a discussion of the implications of these findings for feature selection in the proposed IDS.

4.4.1 Experimental Setup and Evaluation Methodology

Two feature configurations were evaluated in these experiments. The reduced feature set proposed by Yang et al. comprises CAN ID, DATA[5], DATA[3], and DATA[1], whereas the proposed full feature set includes CAN ID together with all payload bytes DATA[0]–DATA[7]. All experiments were conducted using the Car Hacking dataset [1], which is the same dataset employed by Yang et al. [156]. Identical preprocessing procedures, RF hyperparameters, and stratified 70/30 train–test splits were applied throughout, with the feature set being the only experimental variable. For reproducibility, Table 4.6 summarises the hyperparameters and their corresponding values used to train the RF model. To assess robustness to data partitioning, each feature configuration was evaluated across 10 independent stratified splits,

Table 4.6 RF hyperparameters for multiclass classification.

Parameter	Value
n_estimators	50
class_weight	balanced_subsample
max_depth	20
min_samples_leaf	5

ensuring consistent class distributions in both the training and testing sets. Performance was evaluated using accuracy, macro-averaged precision, macro-averaged recall, and macro-averaged F1-score, which provide a balanced assessment in the multiclass setting. Rather than focusing solely on average performance, this analysis explicitly examines split-to-split variability. In addition to performance metrics, permutation feature importance was used as a complementary analysis to quantify the contribution of individual CAN bus features to the RF classifier. For each train–test split, feature importance was computed on the test set by measuring the change in classification accuracy after permuting each feature independently. This procedure was applied consistently across all splits, enabling a systematic assessment of feature importance stability without altering the underlying model or training process.

4.4.2 Consistency Across Train–Test Splits

Figure 4.2 illustrates the performance of both feature configurations across 10 stratified train–test splits. Although both the full and reduced feature sets achieve high performance, as expected under supervised learning, the results demonstrate that the full feature set maintains consistently strong performance across all evaluation metrics, with only minor variation between splits. In contrast, the reduced feature set exhibits noticeably greater fluctuation, particularly in precision and F1-score, indicating increased sensitivity to the specific composition of the training and testing data. This behaviour suggests that while the reduced feature set can achieve strong performance on individual splits, it is less stable and consistently underperforms across all metrics when compared with the complete payload feature set.

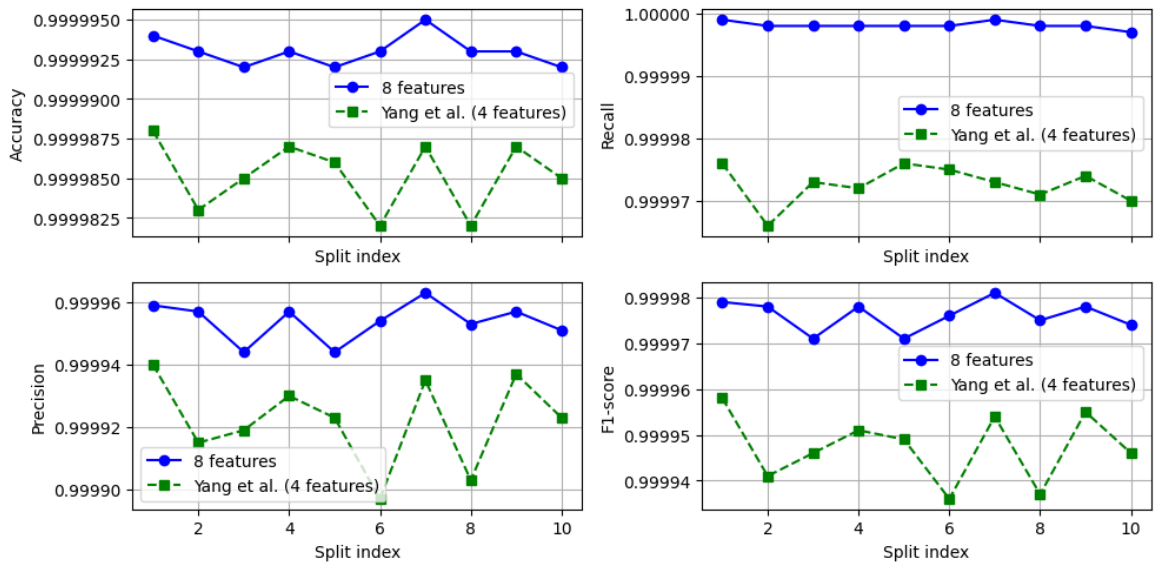


Fig. 4.2 Split-to-split performance of the RF classifier across different data splits.

4.4.3 Robustness and Performance Variation

To provide additional insight into performance variability across data partitions, Figure 4.3 presents boxplots summarising the minimum, median, and maximum values of each metric across all splits. The full feature configuration demonstrates a tighter distribution for all metrics, reflecting robust behaviour even under less favourable partitions. Although the reduced feature set attains comparable maximum values, its lower minimum scores reveal a susceptibility to performance degradation when informative payload features are excluded. Such variability is undesirable in practical IDS deployments, where training data may differ significantly across vehicles and operating conditions.

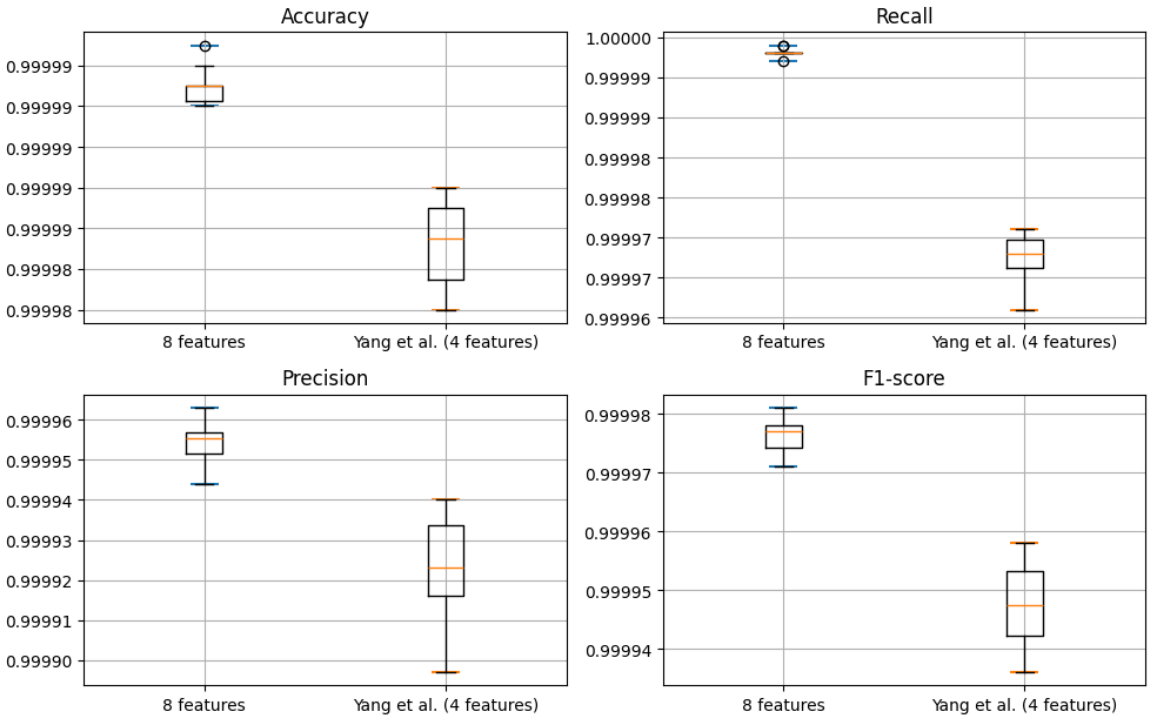


Fig. 4.3 Distribution of RF performance across 10 stratified 70/30 train–test splits.

4.4.4 Feature Importance Stability Across Splits and Datasets

Figure 4.4 shows that feature importance varies across splits. For example, in splits 8 and 9, DATA[6] is the most important feature, whereas in split 5 it is only the third most important feature. This variability indicates that the relative contribution of features is not stable across different train–test partitions. Moreover, DATA[4] is consistently among the least important features in all splits of the first dataset. In contrast, in the second dataset, as shown in Figure 4.5, DATA[4] becomes the first or second most important feature. This observation supports our claim that using all payload features is necessary: although feature importance may vary slightly across splits within the same dataset, it can differ substantially across different CAN bus datasets.

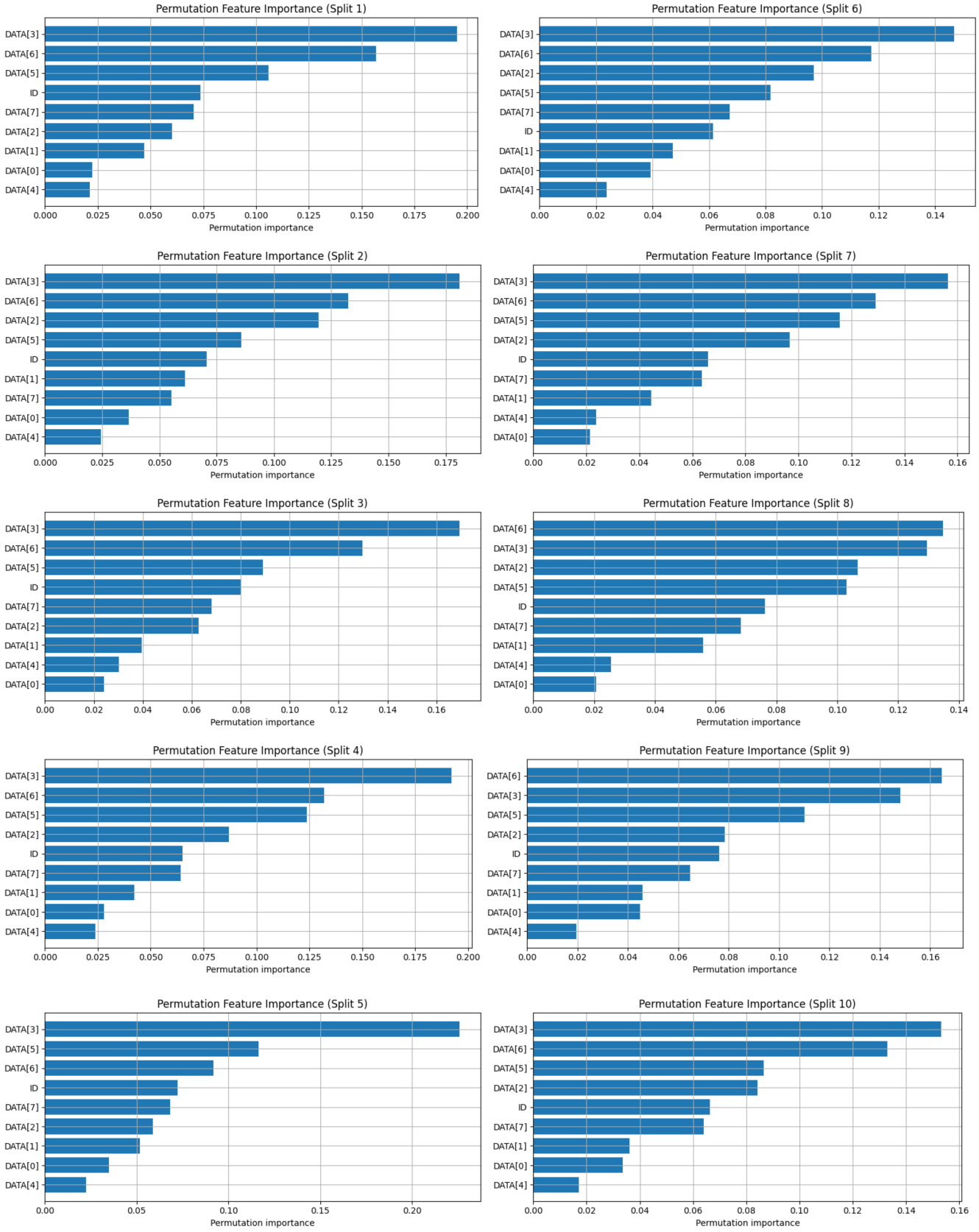


Fig. 4.4 Permutation feature importance across 10 stratified splits for Car Hacking Dataset [1].

Fig. 4.5 Permutation feature importance across 10 stratified splits for Car Hacking: Attack & Defence Challenge 2020 [2].

4.4.5 Interpretation and Implications

These results demonstrate that using the complete payload feature set yields more stable and reliable intrusion detection performance than relying on a reduced set of features. Payload features excluded in the reduced configuration play an important role in sustaining detection performance under varying data distributions, which accounts for the increased performance variability observed when these features are omitted. In addition, the feature importance analysis reveals that feature relevance is not stable: important features vary across train–test splits within the same dataset and can differ substantially across distinct CAN bus datasets. Together, these findings motivate the use of the complete feature set in the subsequent IDS design stage. Retaining all payload features therefore provides a more robust feature representation and supports more consistent performance across different train–test splits, whereas the reduced feature approach is more sensitive to data partitioning and feature importance variability.

4.5 Conclusion

The aim of this chapter was to assess whether selecting specific features of the CAN bus or using all features is more effective in building a robust IDS. Analysis of CAN bus data shows that each CAN ID has unique data patterns, meaning that important features for one CAN ID may not be relevant for another. This variability makes it impractical to select a consistent set of important features across all CAN IDs. Additionally, nearly all comparable feature selection methods identify some or all zero-constant features as important, which are intended to handle irrelevant information. This suggests that applying feature selection techniques uniformly to all CAN IDs may lead to the loss of valuable information.

Although feature selection can create more efficient models, it introduces the risk that attackers might exploit features not considered during the model's training process [163]. For example, a recent study [4] demonstrates the ability to manipulate various CAN bus data and evade the detection model, resulting in false positives. This presents a critical limitation in CAN bus data: selecting a subset of payload features as important while discarding others could allow attackers to leverage neglected features and bypass the model [164, 165]. Furthermore, the evolving landscape of attack scenarios means that features chosen to detect one category of attack may become outdated or insufficient to address new, unseen attacks [163]. Moreover, this chapter demonstrated that retaining all CAN payload features results in more stable and robust intrusion detection performance than relying on a reduced feature set.

Comparative experiments against a widely cited reduced-feature approach showed that the full feature configuration consistently achieved higher and less variable performance across 10 different train–test splits, highlighting its reduced sensitivity to data partitioning. These findings reinforce the argument that preserving complete payload information provides a more reliable foundation for the next chapter on designing a robust in-vehicle IDS. This work contributes to the field of vehicular security by providing a deeper understanding of CAN bus data, supporting the development of IDS models that focus on the most significant features, potentially improving generalisation and enhancing vehicle security. How can the insights gained from focusing on the most significant features of CAN bus data be applied to design a robust in-vehicle IDS that enhances both generalisation and vehicle security? The next chapter will explore this challenge in detail.

Chapter 5

A Robust Multi-Stage Intrusion Detection System for In-Vehicle Network Security [1]

The CAN bus is vulnerable to various cyberattacks, yet it operates under resource constraints. The literature review has highlighted numerous limitations in existing IDS designs, the algorithms employed, and their deployment strategies. Based on our findings in Chapter 4, we decided to include CAN frame features as inputs as the first step in designing an IDS. To enhance the security of in-vehicle networks, this chapter proposes a novel, lightweight, and robust in-vehicle IDS that utilises DL algorithms to detect both known and previously unseen attacks. This IDS not only addresses current and known attacks but is also designed to detect future attacks. We evaluated the proposed IDS using two real-world benchmark datasets. The proposed IDS attains an F1-score of 0.99 for known attacks, while achieving an F1-score above 0.95 and a Detection Rate (DR) of 99.99% for previously unseen attacks, while maintaining a lightweight model size of 2.98 MB and a latency of less than 0.04 ms, meeting the real-time requirements of vehicle safety services.

5.1 Introduction

In Chapter 3, we reviewed the existing literature on in-vehicle IDSs that utilised ML and DL approaches, identifying several research gaps and shortcomings that highlight the urgent need to strengthen the security of these networks. In particular, no proposed IDS can robustly detect and classify known attacks, which is essential for selecting appropriate countermeasures and conducting post-attack analysis [28], while also detecting previously unknown attacks to stay ahead of attackers using DL algorithms with CAN frame features, and at the same

[1] This chapter is published in *Vehicular Communications* Journal 49 (2024): 100837.

time meeting in-vehicle deployment requirements in terms of memory size and deployment environment. These gaps provide a critical foundation for the design of our proposed IDS. Consequently, this chapter aims to develop a robust in-vehicle IDS using DL algorithms that detects both known and previously unknown attacks; while ensuring it remains lightweight in terms of model size to meet resource constraints. Building on the limitations identified in detail in Chapter 3 regarding the design of in-vehicle IDSs, we propose a novel, lightweight, multi-stage IDS tailored for in-vehicle network security, capable of detecting both seen and unseen attacks. This chapter aims to answer the following research questions:

RQ3: *How can an in-vehicle IDS be designed to detect both known and unknown attacks using DL algorithms?*

RQ4: *How can the IDS remain lightweight and meet real-time requirements while operating within resource constraints?*

In answering these questions, the following contributions are made:

C3 To propose a novel, lightweight in-vehicle IDS that leverages DL algorithms for efficient and practical deployment.

C4 To design the IDS to detect and classify known attacks while also identifying previously unseen and adversarial ones.

This work 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. 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. 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 utilise a hybrid approach in in-vehicle IDSs to detect both seen and unseen attacks using DL algorithms, rather than traditional ML algorithms, while achieving a small model size.

5.2 Related Work

In Chapter 3, we provided a more extensive discussion of related work and formally established the research question that underpins this study. However, in this section, we

offer a concise overview of similar studies, examining their limitations in greater depth and demonstrating how our approach distinguishes itself from existing research.

5.2.1 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. As discussed in Chapter 3, researchers have explored a wide range of approaches to developing these systems. However, few papers have developed an in-vehicle IDS capable of identifying both seen and unseen attacks [1, 63, 155–161]. These studies and their limitations are summarised in Table 5.1, highlighting our contribution.

Table 5.1 Related works of ML-based IDSs for in-vehicle network.

Ref	Year	Category	Algorithm	Dataset	M-C	ID-based Detection	Payload-based Detection	DL	FL
[63]	2022	Semi-supervised	AE, GAN	Car-Hacking [1]		✓		✓	
[1]	2018	Unsupervised	GAN	Car-Hacking [1]		✓		✓	
[155]	2019	Supervised	DNN	Simulation			✓	✓	
[157]	2021	Unsupervised	LightGBM, AE	Survival Analysis Dataset for automobile IDS [122]		✓	✓	✓	
[158]	2024	Semi-supervised	BNN, GAN	Car Hacking [1], Survival Analysis Dataset for automobile IDS [122]	✓	✓		✓	
[159]	2021	Unsupervised	DT, RF, XGBoost	Own	✓	✓			
[160]	2020	Supervised/ Unsupervised	FCN, CNN, TCN, LSTM, AE	SynCAN [146]			✓	✓	
[161]	Nov-2024	Semi-supervised	VAE and AERL	car-hacking [85], ROAD [119]	✓	✓	✓	✓	
[156]	2022	Hybrid	DT, RF, ET, XGBoost, CL-k-means	Car-Hacking [1], CICIDS2017 [168]	✓	✓	✓		
Our work	**Aug-2024**	**Hybrid**	**ANN, LSTM-AE**	**Car-Hacking** [1]	✓	✓	✓	✓	✓

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

A similar approach is introduced by Yang et al. [156], who 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. Building on this work, our study contributes to vehicular security by overcoming several of the constraints identified in MTH-IDS [156]:

- **Comprehensive use of CAN frame features:** Our study contributes by moving beyond the restricted feature selection adopted in earlier works such as MTH-IDS,

which relied on only four features—CAN ID, DATA[5], DATA[3], and DATA[1]—to train the model. Selecting such a small subset of features leaves the possibility for attackers to manipulate fields that were not considered during training [163]. By contrast, our approach utilises the full CAN frame, which is crucial for three reasons. First, it reduces the risk of attackers exploiting neglected fields and bypassing the model [164, 165]. Second, it ensures adaptability to the evolving landscape of attack scenarios, since features effective against one type of attack may not remain sufficient for detecting new, unseen threats [163]. Third, as demonstrated in Chapter 4, each CAN ID exhibits distinct patterns in its data fields. Consequently, features that are significant for one CAN ID may not be relevant for another, making it impractical to rely on a fixed subset across all CAN IDs. By accounting for these characteristics, our contribution lies in enabling IDS models that exploit the full CAN frame to capture the most meaningful features, thereby improving robustness and generalisation capability.

- **Adoption of DL methods:** Our study contributes by employing DL algorithms instead of conventional ML models, as used in MTH-IDS. Prior work has shown that DL-based IDSs outperform ML-based approaches in automotive applications [166]. This superiority is due to several factors: DL methods are more adaptive, continually refined with incoming data, which is particularly suitable for the nature of CAN bus data [155]. Unlike traditional ML, which often depends on manual feature engineering, such as correlation-based feature selection, a process that is both time-consuming and limited in scope, DL automatically learns and extracts the most relevant features directly from raw data [167]. Moreover, DL-based IDSs demonstrate superior capability in detecting new and unknown attacks and can scale more effectively to highly complex in-vehicle network data while maintaining performance [167]. By adopting DL in our design, we contribute a more robust and effective IDS that overcomes the constraints of traditional ML approaches in the automotive domain.

- **Avoidance of biased classifiers:** Our study contributes by designing an IDS that does not rely on biased classifiers trained on specific errors such as False Positives (FPs) and False Negatives (FNs), as introduced in MTH-IDS. While such classifiers may improve performance on a given dataset, they often degrade when applied to new, unseen data. Moreover, their use transforms the model from being unsupervised into one that depends on labelled datasets, which are difficult to obtain and implement in real vehicular environments. By avoiding this dependency, our approach maintains stronger generalisability and practicality for deployment in in-vehicle networks.

- **Double layer of protection:** Our study contributes by introducing a double layer of protection that detects both known and unknown attacks while also re-examining data initially classified as normal. Although a few existing works have proposed using two models, none have incorporated this re-examination step. This is important because attackers constantly seek ways to evade detection models, and incorporating a second model not only to detect new and unknown attacks but also to double-check the normal data can provide an additional layer of protection. According to recent studies [4, 179], in-vehicle IDSs are vulnerable to evasion attacks, where attackers manipulate input data to deceive models into producing incorrect or misclassified outputs [179], posing significant risks to the safety and security of CAVs.

- **Deployment approach:** Our study contributes by avoiding the reliance on server-based training and instead designing an IDS deployment strategy that does not require transmitting private CAN data to a remote server. As shown in Table 5.1, previous studies [1, 63, 155–161] proposed training on a server machine and testing on the CAN bus, which introduces privacy concerns due to the need to send private data externally. In addition, attackers can manipulate transmitted data to degrade a model's performance, resulting in misleading predictions [180]. By keeping training and deployment within the vehicle environment, our approach mitigates these risks and contributes to a more secure and practical IDS implementation.

While previous studies have achieved notable results in specific areas, they also have several limitations as highlighted in Chapter 3 section 3.4.3. Compared to existing studies on in-vehicle IDS, our proposed IDS contributes to the field of vehicular security in several ways: 1) It uses DL rather than traditional ML, which has proven to be more efficient in automotive applications and is capable of detecting new unknown attacks more effectively [166, 167]. 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 [181, 35, 181, 103]. 3) It successfully detects both seen and new, unseen attacks. 4) Despite using DL algorithms, the IDS remains lightweight, operating within the typical memory capacity of vehicle-level machines, which can exceed 1 GB of RAM [156]. 5) It utilises both CAN ID and payload as features, which enables detection of CAN ID changes and payload manipulation attacks [27]. 6) It consists of two cascaded models: the first model classifies traffic into seen attacks and normal data, while the second model re-evaluates the data classified as normal by the first model to detect any tweaks intended to bypass the initial classification, thereby enhancing the

robustness of the IDS. 7) It continuously learns by labelling and updating the signature-based classifier when a new, unseen attack is detected.

The novelty of our IDS 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. We adopted the seen attack detection before anomaly detection approach to enable rapid detection and response to known attacks, and to double-check normal data in the event an attack evades the first model, thus reducing the risk of evasion attacks. 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 labelling 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 defence mechanism against potential attacks and improves detection performance compared to single-point IDSs [182].

5.3 Methodology

In this section, we present the methodologies used to develop our proposed in-vehicle IDS. First, we describe the experimental setup. Second, we explain the proposed IDS, including a description of the datasets, data pre-processing, the first stage (a supervised model using ANNs), and the second stage (an unsupervised model using an LSTM-autoencoder). Finally, we provide details on constructing the model, including hyperparameter tuning for both the ANN and the LSTM-autoencoder and the generation of adversarial samples.

5.3.1 Experimental 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.

5.3.2 Proposed Novel In-Vehicle IDS

In Section 5.2, 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 [35, 181, 103]. 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 Figure 5.1, the CAN bus data first enters the data pre-processing 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 categorising 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 are sent to security centres for analysis and verification by security experts. If these anomalies are confirmed as attacks, a new attack label is generated. This label will then be used to retrain and refine the supervised model, enabling it to recognise similar attacks in the future and ensuring continuous improvement of the system. Thus, 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 behaviour for each vehicle at design time and monitor the network to detect any attacks during operational runtime.

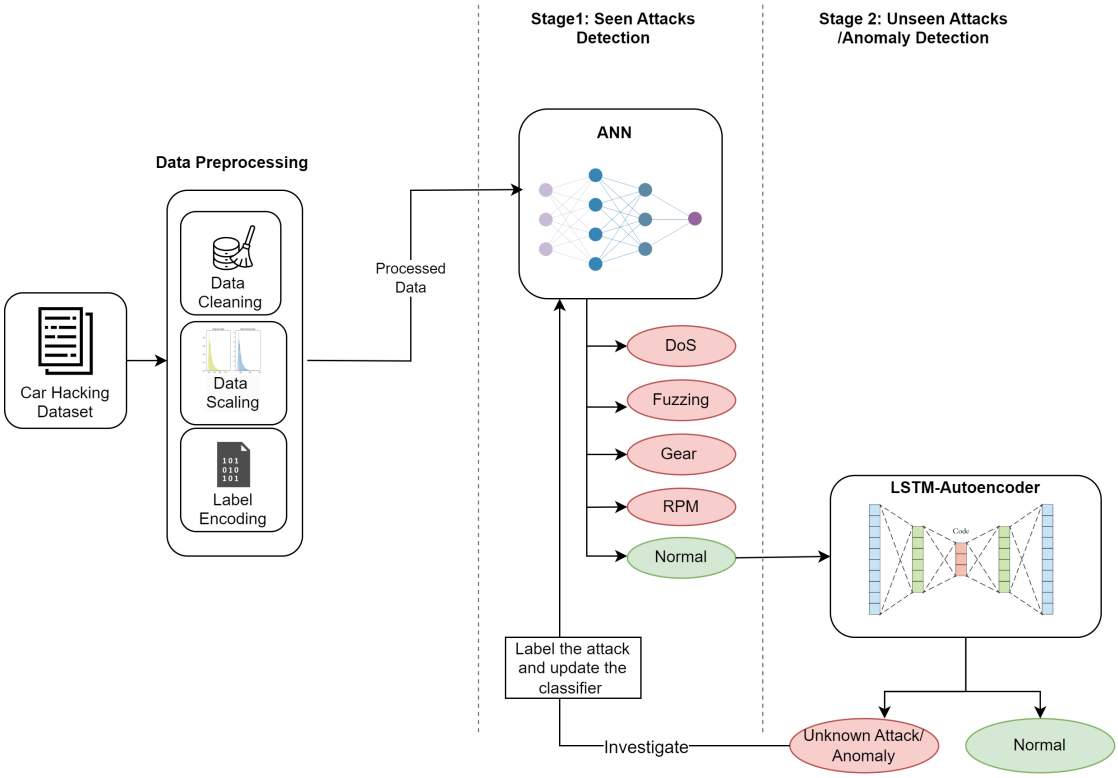


Fig. 5.1 Workflow model depiction of proposed multi-stage IDS for in-vehicle network.

Our proposed IDS provides a comprehensive and robust solution to secure in-vehicle networks. 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 attacks.

Datasets Description

To assess the performance of our proposed model, we used two benchmark datasets. The primary dataset used in this thesis is the Car Hacking Dataset published by Song et al. [1]. This dataset is widely used in automotive security research and includes four types of attacks: DoS, frame fuzzification, engine RPM spoofing, and drive gear spoofing. We selected it because it is based on real-world traffic data rather than simulated data. Moreover, its status as the most frequently cited resource in the field's literature [27] allows for direct comparison of our proposed IDS approach with related work using the same dataset [156]. Additionally, we assess the performance of our proposed IDS using another dataset, the Car Hacking: Attack & Defence Challenge 2020 dataset [2], which is widely recognised as a benchmark in automotive security research. This dataset comprises normal traffic and four types of attacks: DoS, frame fuzzification, replay and spoofing [2]. However, replay attacks were excluded

from this work as they exploit message timing rather than content, by re-injecting identical CAN IDs and payloads at different time points. Detecting such attacks requires temporal analysis, which is beyond the scope of this work. Since CAN IDs may follow periodic or event-driven transmission patterns, based on specific time intervals or triggered events [27], accurate detection relies on detailed timing specifications. These specifications are defined in the DBC file, which is proprietary to vehicle manufacturers.

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 5.2, we present the dataset's features, descriptions, and data types.

Table 5.2 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

To select the most suitable algorithms, we examined the CAN bus data from Car Hacking Dataset [1], ensuring a comprehensive understanding of both normal and abnormal behaviours. Below is a brief description of the attacks, including Frame (DoS, Fuzzification, RPM, and Gear Spoofing Attacks), which are explained in detail in Chapter 2, Section 2.7.

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.

Table 5.3 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

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

Data Pre-processing

During data pre-processing, the dataset was transformed into a format suitable for DL algorithms through a series of operations. Figure 5.2 illustrates the processing steps applied to each feature. The Car Hacking Dataset [1] comprises four files, each corresponding to a specific attack type: DoS, frame fuzzification, gear spoofing, and RPM spoofing. Instances were flagged as *T* for attack and *R* for normal data. Table 5.4 summarises the number of attack and normal instances in each file.

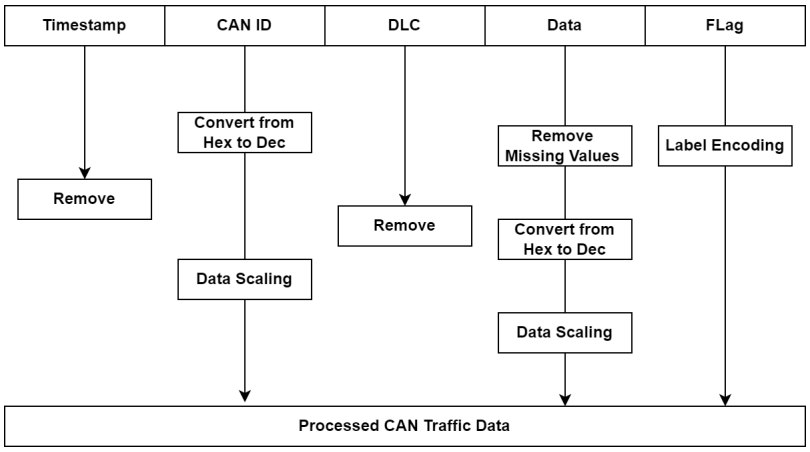


Fig. 5.2 Data pre-processing.

Table 5.4 Car Hacking dataset [1] 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%)

Each file was preprocessed individually before being merged into a single dataset. To avoid mixing different attack types during concatenation, T values in the Flag column were replaced with the corresponding attack names, such as DoS, Fuzzy, RPM, and Gear, and R was replaced with Normal. Both the CAN ID and data fields were originally in hexadecimal format and were converted to decimal values to meet the input requirements of ML models. A key issue identified in the dataset was the misplacement of labels when the DLC value was less than 8. In such cases, only a subset of data fields (for example, DATA[0] to DATA[3]) is filled, while the remaining fields (for example, DATA[4] to DATA[7]) should be empty. However, labels were incorrectly inserted into the first available empty data field. To address this and adjust the label misplacement, we developed a Python function that checks each row's DLC value, moves misplaced labels to the appropriate label column, and replaces the unused data fields with NaN values. Rows with missing values were subsequently removed

due to the large volume of available data. We also dropped the Timestamp and DLC features. The Timestamp was excluded as it is generated by the CAN logger and is not an intrinsic part of the CAN frame [4], and is therefore typically omitted in IDS research unless specifically needed. The DLC feature, which indicates the length of the data field, was removed due to its strong correlation with the data fields and to reduce feature dimensionality [27]. Since most ML algorithms require numerical inputs, categorical values in the Flag column (such as Normal, DoS, frame fuzzification, gear spoofing, and RPM spoofing) were converted into numeric format using a label encoder after all datasets were merged. Moreover, the same pre-processing steps were applied to the Car Hacking: Attack & Defence Challenge 2020 dataset [2], except for minor differences in issues related to the dataset itself. Table 5.5 summarises the number of attack and normal instances in the dataset. Moreover, Algorithms 1 and 2 provide a detailed overview of the pre-processing steps in the exact order in which they were applied to each dataset.

Table 5.5 Car Hacking: Attack & Defence Challenge 2020 dataset [2] overview.

Data type	**No of Instances**
Flooding	345,859
Frame Fuzzification	186,572
Spoofing	50,339
Normal	6,730,953
Total	7,313,723

Algorithm 1 Detailed data pre-processing steps for Car Hacking dataset [1].

Input: Dataset D with four files (DoS, Fuzzification, Gear, RPM), each containing attack and normal instances

Features: Timestamp, CAN ID, DLC, DATA[0] to DATA[7], Flag

Output: One processed dataset D'

Begin

1: Load files f_i
2: **for** each file f_i in dataset D **do**
3: Convert CAN ID from hexadecimal to decimal using *pandas.Series.apply()*
4: Convert T in the *Flag* column to the corresponding attack name and R to Normal
5: Move *Flag* to the last column
6: Fill missing data bytes with `NaN` values
7: Remove rows with missing values
8: Drop *Timestamp* and *DLC* features
9: Convert *data field* values from hexadecimal to decimal using *pandas.Series.apply()*
10: **end for**
11: Concatenate all files into a single dataset D'
12: Use *LabelEncoder()* to convert categorical *Flag* values into numeric labels: (DoS: 0, Fuzzy: 1, Gear: 2, Normal: 3, RPM: 4).

End

Algorithm 2 Detailed data pre-processing steps for Car Hacking: Attack & Defence Challenge 2020 dataset [2]

Input: Dataset D with eight files: Two files from the submission data: `Pre_submit_D.csv` and `Pre_submit_S.csv`, and six files from the training data: `Pre_train_D_0.csv`, `Pre_train_D_1.csv`, `Pre_train_D_2.csv`, `Pre_train_S_0.csv`, `Pre_train_S_1.csv`, and `Pre_train_S_2.csv`.

Features: Timestamp, Arbitration_ID, DLC, Data, Class, SubClass (not present in files containing only normal data)

Output: One processed dataset D'.

Begin

1: Load the eight files f_i
2: If *SubClass* not in f_i, create it and assign all values as Normal
3: Merge f_i files into a single dataset file
4: Rename columns: *Arbitration_ID* to *CAN ID* and *SubClass* to *Flag* for consistency
5: Convert *CAN ID* values from hexadecimal to decimal using *pandas.Series.apply()*
6: Drop unnecessary columns: *Timestamp*, *DLC*, and *Class*
7: Split the *Data* field into separate columns: *DATA[0]* to *DATA[7]*
8: Delete the original *Data* field and move the *Flag* column to the end
9: Remove rows with missing values
10: Convert *DATA* field values from hexadecimal to decimal using *pandas.Series.apply()*
11: Delete any row with the Replay label in the *Flag* column
12: Use *LabelEncoder()* to convert the categorical *Flag* feature into numeric labels: (Flooding: 0, Fuzzing: 1, Normal: 2, Spoofing: 3)

End

As shown in Table 5.4, 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 [183]. To address the issue of class imbalance, we use the Synthetic Minority Oversampling Technique (SMOTE), which is widely used for handling data imbalance, particularly in in-vehicle data, as shown in [156, 147, 105]. Figure 5.3 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 generalised and capable of recognising the entire dataset. We

used this method only for unsupervised anomaly detection because the supervised classifier handled the entire dataset efficiently and quickly.

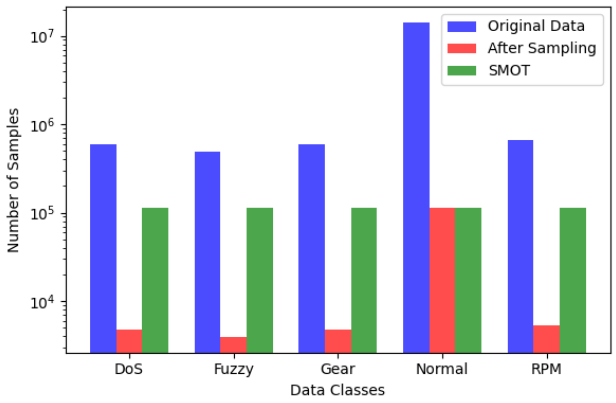


Fig. 5.3 Comparison between the number of data classes before and after sampling and balancing data.

We considered nine columns (CAN ID and eight data values) as the features in our experiments. This choice was made to detect any manipulation of either the CAN ID or the payload. We then applied the StandardScaler technique to scale the data, which helps find a better balance between handling outliers and preserving the range of values. It standardises 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 (5.1):

$$X_{\text{scaled}} = (X - X_{mean})/X_{std} \tag{5.1}$$

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 normalised (scaled) datasets tend to enhance the performance of ML models [184]. Failure to normalise the dataset, especially when its features have different scales, may cause the ML model to concentrate primarily on the features with larger scales [156]. Table 5.6 shows the data features and types before and after data pre-processing.

Table 5.6 Data features and types before and after data pre-processing.

Feature	Before	After
CAN ID	hexadecimal	decimal integer
DATA[0]	hexadecimal	decimal integer
DATA[1]	hexadecimal	decimal integer
DATA[2]	hexadecimal	decimal integer
DATA[3]	hexadecimal	decimal integer
DATA[4]	hexadecimal	decimal integer
DATA[5]	hexadecimal	decimal integer
DATA[6]	hexadecimal	decimal integer
DATA[7]	hexadecimal	decimal integer
Flag	string (T, R)	decimal integer (0,1,2,3,4)

First Stage: Supervised Model

The first stage of the proposed multi-stage IDS is a supervised model responsible for detecting and classifying seen attacks.

Artificial Neural Networks

Artificial Neural Networks (ANNs) are ML algorithms inspired by the behaviour of biological neurons in the brain and the central nervous system [185, 186]. ANNs' inputs pass through one or more hidden layers, assign weights, and produce an output. Figure 5.4 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 [187]. Therefore, an ANN can identify relationships within complex datasets [7]. 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 [188]. Their adaptability, particularly in high-dimensional datasets,

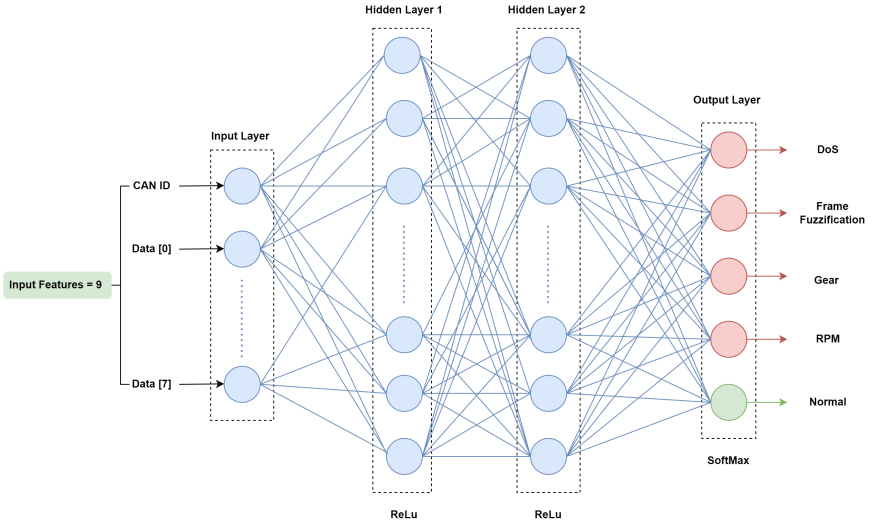


Fig. 5.4 ANN architecture.

addresses challenges associated with conventional algorithms such as k-nearest neighbor and decision trees [189]. 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 [190, 191].

We opted to use ANNs for multiclass classification for the following reasons:

- ANNs introduce non-linearity into the model, which makes them capable of modeling complex patterns and relationships that linear classifiers might miss.
- Unlike traditional methods, ANNs can automatically learn the best features directly from the raw data without requiring explicit feature engineering [167], which is often time-consuming [69]. This can lead to better performance, especially in CAN bus data, where each CAN ID has distinctively informative features.
- ANNs are adaptive systems, making them suitable in situations where data is continuously evolving [192].
- DL algorithms, including ANNs, demonstrate their superior ability to process large volumes of data in a faster and more efficient manner [42]. As for the CAN bus, the ECUs in a modern vehicle generate about 2,000 CAN frames per second [1].
- Based on a comparison with traditional ML algorithms—Decision Tree (DT), Random Forest (RF), Naive Bayes (NB), k-nearest Neighbors (KNN), and XGBoost (Extreme

Table 5.7 Classifiers comparison.

	ANN	DT	RF	NB	KNN	XGBoost
Accuracy	99.99	100	100	99.36	100	100
F1-Score	1.0	1.0	1.0	0.97	1.0	1.0
Precision	1.0	1.0	1.0	0.99	1.0	1.0
Recall	1.0	1.0	1.0	0.95	1.0	1.0

Gradient Boosting), as shown in Table 5.7, ANNs can produce highly competitive results.

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 [127]. 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 labelled 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 [63, 134, 32]. 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.

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 [27]. 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 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 [27]. 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. Figure 5.5 depicts the architecture of the LSTM-autoencoder. A similar use of an LSTM-autoencoder in a network IDS is demonstrated in [125], though applied within the context of a software-defined networking environment.

5.3.3 Constructing the Model

To prevent overfitting in ML models, we applied the data partitioning method outlined in [193] by dedicating 70% to training and the subsequent 30% to testing the model's performance. In the ANN model, we trained the model using labelled 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 5.3.2. The data classified

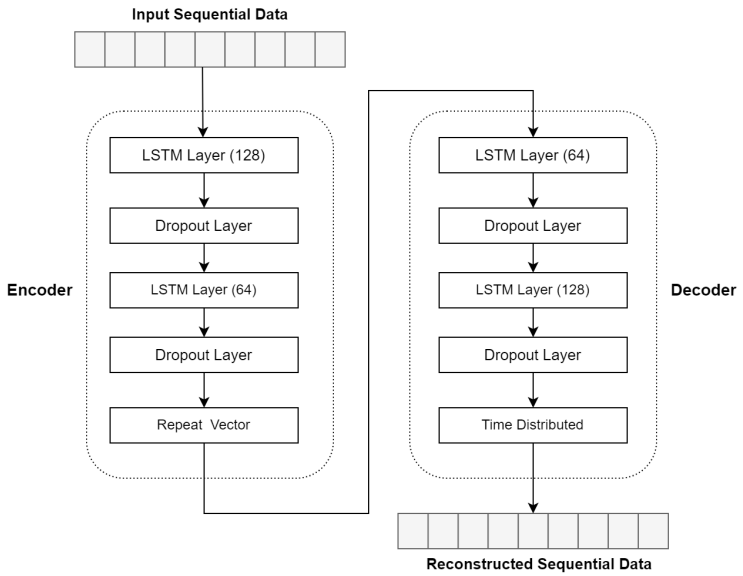


Fig. 5.5 LSTM-Autoencoder architecture.

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 hyperparameters, such as batch size, number of units, learning rates, optimisers, activation, and loss functions. Through systematic experimentation, we identified the most suitable hyperparameter values that could deliver the best possible performance and efficiency.

5.3.4 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). To keep the algorithm as simple as possible and ensure it remains lightweight, we used a sequential model to structure our neural network [9]. We employed the GridSearchCV algorithm [194] from scikit-learn 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 utilising 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 optimised

the model with the Adam optimiser, maintaining the default values for other parameters. To enhance training efficiency and mitigate overfitting, we integrated an EarlyStopping callback. Table 5.8 summarises the hyperparameters and their respective values used in tuning the ANN model.

Table 5.8 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
Optimiser	Adam
Batch Size	256
Shuffle	True
Loss Function	categorical_crossentropy

5.3.5 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.

Table 5.9 shows the parameters and their respective values in hyper-tuning the LSTM-autoencoder. We trained the LSTM-autoencoder using only the normal data labelled 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

Table 5.9 LSTM-Autoencoder parameters for binary classification.

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

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 that 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 (5.2):

$$\text{Threshold} = \mu(\text{train\_errors}) + \sigma(\text{train\_errors}) \tag{5.2}$$

In this context, μ represents the mean, while σ indicates the standard deviation. Figure 5.6 presents a concrete numerical example illustrating the data flow through the LSTM-autoencoder, from the original CAN message to the final output. The raw CAN data is first scaled and reshaped from a two-dimensional format of (Samples, Features) into a three-dimensional format of (Samples, Time Steps, Features) to satisfy the LSTM input requirements. The autoencoder then reconstructs the input sequence, and the reconstruction error between the original and reconstructed samples is computed as the mean squared error between the input and reconstructed sequences. This reconstruction error serves as a continuous anomaly score, which may take values such as 0.004, 0.12, or 3.55. A predefined threshold of 0.02 is then applied to convert this score into a binary decision, whereby samples with reconstruction errors exceeding the threshold (e.g. 0.12 or 3.55) are classified as attacks, while those below the threshold (e.g. 0.004) are classified as normal traffic.

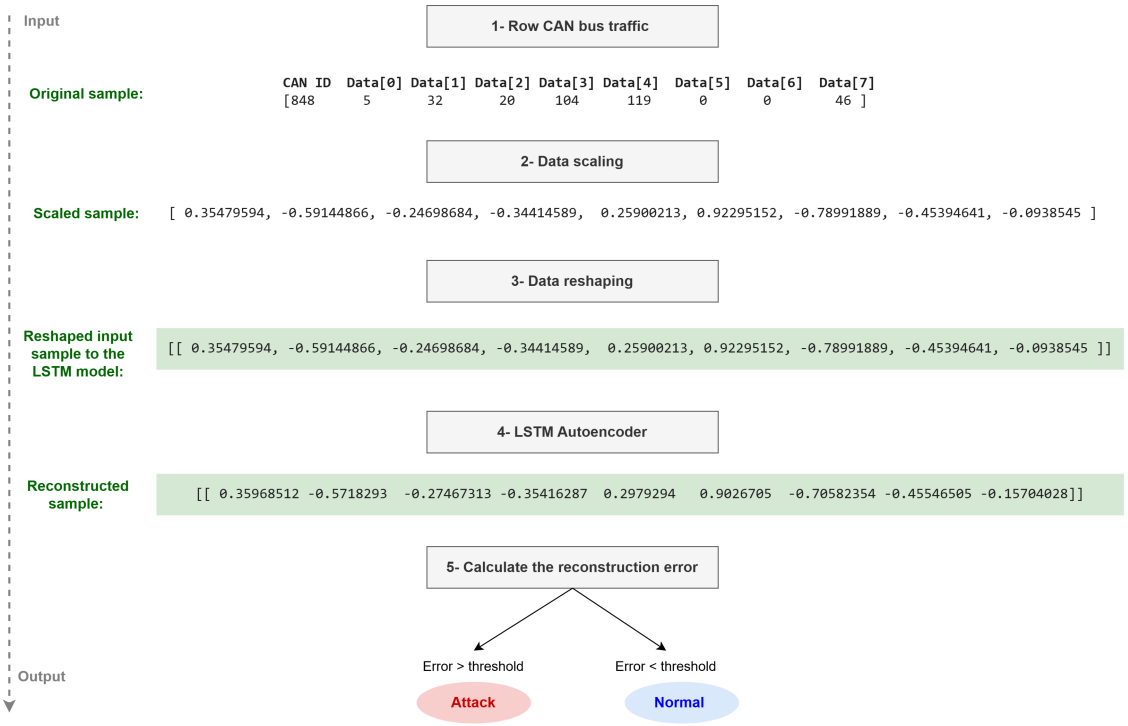


Fig. 5.6 Illustrative example of LSTM-autoencoder input, reconstruction, and anomaly decision for a single CAN sample.

5.3.6 Generation of Adversarial Samples

ML- and DL-based IDSs have demonstrated vulnerability to adversarial attacks. These attacks involve manipulating input data to deceive models into generating incorrect or misclassified outputs [179]. Such vulnerabilities pose significant risks to the safety and security of CAVs.

To assess the robustness of our proposed in-vehicle IDS against such attacks, we adopted the methodology proposed by [4] for generating adversarial samples and used it to evaluate our IDS's resilience and training.

Given the simple structure and low dimensionality of the CAN frame, only a limited number of features are subject to manipulation. In particular, the data field, which consists of eight dynamic bytes, is the only feature that can be realistically altered, thereby providing adversaries with a potential avenue for exploitation. To ensure that only the eight bytes of the data field are manipulated, a boolean mask is applied during the adversarial example generation process, where only the data field bytes are marked 'True' for manipulation, while all other fields are excluded.

Among several adversarial methods, this thesis adopts the Basic Iterative Method (BIM) to generate adversarial examples. According to [4], BIM has proven to be one of the most effective adversarial attack methods for bypassing targeted IDSs.

The implementation of BIM provided by the Adversarial Robustness Toolbox (ART) takes the IDS model, the original samples, and a specified target label as input. In this context, the target label is defined such that normal samples are mislabelled as malicious, and malicious samples as normal. The purpose is to trick the IDS into making incorrect classifications.

The generation of these adversarial examples begins with the original, unaltered samples. These examples are iteratively refined, with the IDS used at each step to assess their effectiveness. The process concludes when a sample is classified as the target label.

The epsilon value represents the magnitude of the perturbation applied to the input data to cause the targeted IDS to misclassify it into a specific class. In [4], various epsilon values ranging from 1 to 10 were tested to examine their impact; however, we set epsilon to 10, as their results demonstrated that larger perturbations generally led to higher attack success rates.

5.4 Results and Evaluations

In this section, we present the evaluation metrics used to assess the performance of the proposed IDS, the performance results for seen, unseen, and adversarial attack detection, the model complexity, and a comparison with existing studies.

5.4.1 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, detection rate (DR)—and False Alarm Rates (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:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.3}$$

$$F1 = 2 \times \frac{Pre \times Rec}{Pre + Rec} \tag{5.4}$$

$$Pre = \frac{TP}{TP+FP} \tag{5.5}$$

$$Rec, DR = \frac{TP}{TP+FN} \tag{5.6}$$

$$FAR = \frac{FP}{TN+FP} \tag{5.7}$$

5.4.2 Performance Results and Analysis of Seen, Unseen, and Adversarial Attack Detection

This subsection summarises the results of our proposed IDS in detecting seen attacks, unseen attacks, and adversarial samples, and offers an analysis of these findings.

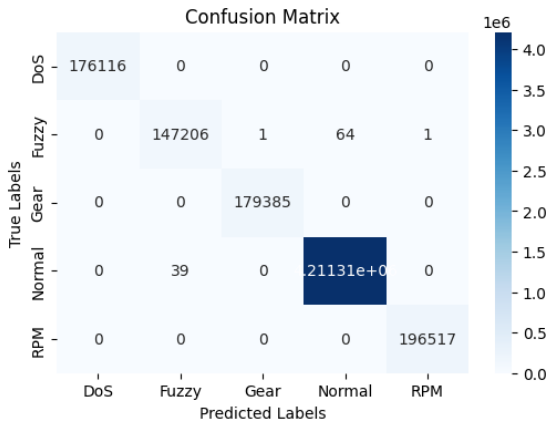
Seen and Unseen Attack Detection

Starting with the results for seen attack detection in the first model, the ANN was trained and tested on a labelled dataset that included normal data and four types of attacks: DoS, frame fuzzification, RPM, and gear spoofing. The performance of the ANN model on the two datasets is detailed in Table 5.10, 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. Figure 5.7 displays the multiclass confusion matrix of the ANN model on the two datasets, which illustrates the model's capability to classify test data into multiple attack categories. Moreover, as depicted in Figures 5.8 and 5.9, the training and validation losses decreased and converged over time, indicating that the models were learning effectively and generalising well without overfitting.

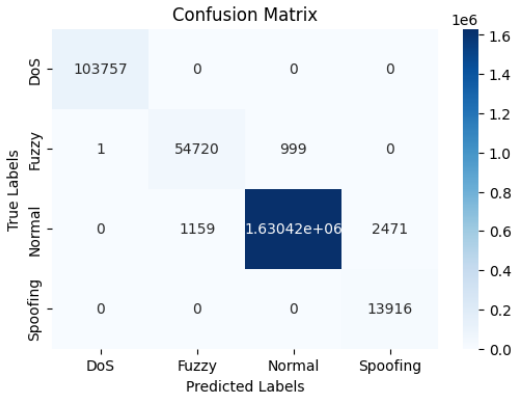
To evaluate the model's ability to detect new, unseen attacks, we trained the LSTM-autoencoder on a sample of normal data and then tested it on the remaining dataset. As shown in Table 5.11, the LSTM-autoencoder performed well, with an overall accuracy of 98.59%, an F1-score of 0.95, a DR of 99.99%, 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 attacks. For each type of unseen attack, the model successfully detected all attacks with a DR between 99.99% and 100%, accuracy

Table 5.10 Performance evaluation of ANN on seen attacks detection.

Attack	Acc (%)	F1	Pre	DR(%)	FAR
Car Hacking Dataset [1]					
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
Car Hacking: Attack & Defence Challenge 2020 Dataset [2]					
DoS	99.99	0.99	0.99	100	0.0000006
Frame Fuzzification	99.88	0.98	0.98	0.98	0.0006
Spoofing	99.86	0.92	0.85	100	0.0013
Normal	99.74	0.99	0.99	99.81	0.0057



(a) Confusion matrix for Car Hacking dataset [1].



(b) Confusion matrix for Car Hacking: Attack & Defence Challenge 2020 dataset [2].

Fig. 5.7 ANN multiclass confusion matrix.

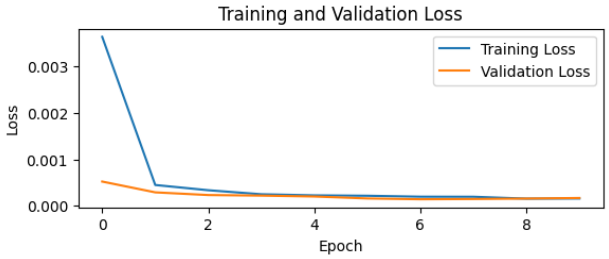


Fig. 5.8 ANN training and validation performance.

exceeding 98%, and a low FAR of 0.016%. However, the F1-score varied from 0.81 to 0.85 for specific types of unseen attacks. Figure 5.10 shows the binary confusion matrix of the LSTM-autoencoder model, illustrating its performance in classifying the test data into

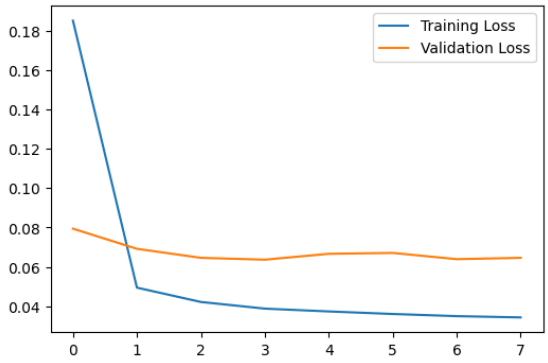


Fig. 5.9 LSTM training and validation performance.

normal (0) and anomaly (1) classes. Figure 5.10a shows the confusion matrix on the test samples after applying SMOT sampling, as described in Section 5.3.2, while Figure 5.10b displays the confusion matrix 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.

Table 5.11 Detection results for unseen attacks for Car Hacking dataset [1].

Unseen Attack	**Number of normal training data**	**F1**	**Pre**	**Acc (%)**	**DR (%)**	**FAR (%)**
DoS	79,809	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

Anomaly detection in the proposed LSTM-autoencoder is based on a thresholded reconstruction error. Each CAN message is assigned a reconstruction error, which is compared against a fixed threshold to produce a binary classification. Based on these thresholded predictions, TPs, TNs, FPs, and FNs are computed by comparing the predicted labels with the ground-truth labels. Table 5.12 provides detailed class distributions and a breakdown of TP, TN, FP, and FN values for each evaluated attack type. Across all attacks, the model achieves perfect or near-perfect detection performance, with zero or only one FN, indicating that malicious traffic is rarely missed. The single missed detection occurs in the frame fuzzification attack, where the anomalous frame is very similar to normal traffic and therefore produces a low reconstruction error. In contrast, most incorrect predictions occur when normal frames are misclassified as attacks, resulting in FPs. To illustrate this behaviour and clarify the input and output of the LSTM-autoencoder in such cases, Table 5.13 presents

Table 5.12 Class distribution and breakdown of TP, TN, FP, and FN for each evaluated attack type.

Attack type	Total testing samples	Normal samples	Attack samples	TP	TN	FP	FN
Frame fuzzification	14,729,805	14,237,958	491,847	491,846	14,003,364	234,594	1
DoS	14,825,479	14,237,958	587,521	587,521	14,003,364	234,594	0
RPM	14,892,855	14,237,958	654,897	654,897	14,003,364	234,594	0
Gear	14,835,210	14,237,958	597,252	597,252	14,003,364	234,594	0

Table 5.13 Illustrative example of normal data misclassified as an attack.

Original normal CAN message	
[2010 3 127 208 128 255 255 255 255]	
Scaled CAN message	
[3.28007174 -0.61369813 1.49746943 2.8803886 0.49410909 2.77925181 2.51898897 4.01603601 2.51863526]	
Reshaped CAN message (LSTM-autoencoder input)	
[[3.28007174 -0.61369813 1.49746943 2.8803886 0.49410909 2.77925181 2.51898897 4.01603601 2.51863526]]	
Reconstructed CAN message (LSTM-autoencoder output)	
[[0.7638385 -0.39255068 -0.51519686 0.6548352 0.05684954 1.9956354 0.5229528 2.2812848 -0.4475813]]	
Reconstruction Error	3.553495
Threshold	0.0223150
Reconstruction Error >threshold	yes (1)
Output	Attack

a detailed example of a normal CAN message that is incorrectly classified as an attack, including the input data, reconstructed output, reconstruction error, and final model decision. As shown, the LSTM-autoencoder struggles to accurately reconstruct the input sequence, producing a high reconstruction error of 3.55, which exceeds the predefined threshold of 0.02. These FPs occur because the LSTM-autoencoder was trained on only a subset of normal traffic rather than the full normal dataset. Consequently, normal patterns that were not observed during training are reconstructed less accurately, leading to higher reconstruction errors and their misclassification as attacks.

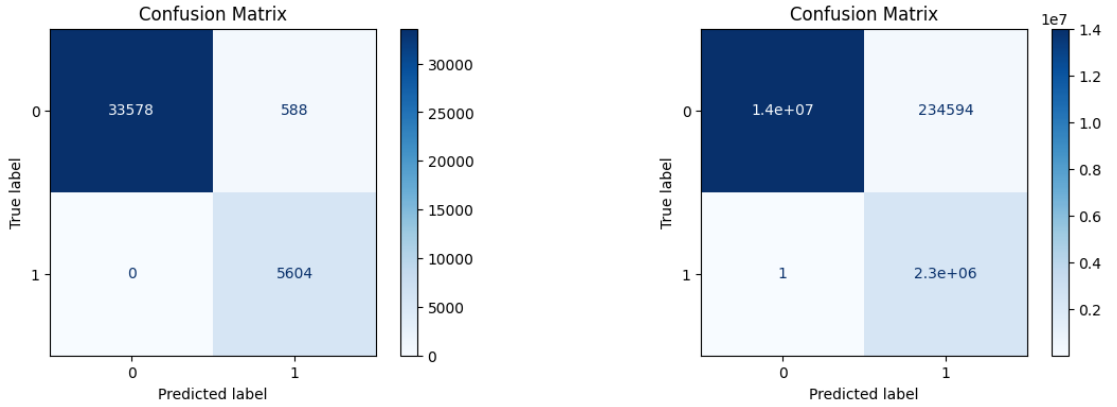
In addition, we evaluated the LSTM-autoencoder on the second dataset [2] for unseen attack detection. The resulting performance is summarised in Table 5.14, which exhibits

lower performance than Table 5.11, with an F1 score of 0.75, a DR of approximately 94%, and a FAR of 0.059. This degradation is primarily attributable to differences in dataset characteristics, as the same model architecture was used in both experiments. The weakest performance in Table 5.14 observed for spoofing attacks, which the model fails to classify as malicious. This behaviour is expected given the high similarity between spoofed and normal CAN messages in this dataset, where attacks differ from benign traffic only through marginal payload variations. For example, a normal CAN frame [1379, 192, 3, 0, 0, 0, 0, 1, 1] and a spoofed frame [1379, 193, 3, 0, 0, 0, 0, 1, 1] differ in only a single payload byte with numerically adjacent values, providing insufficient discriminative information for reliable detection. This explains the low spoofing detection values reported in Table 5.14. In contrast, the dataset used in Table 5.11 contains spoofing messages that, while sharing the same CAN ID as normal traffic, exhibit substantial differences across multiple payload bytes. For instance, a normal CAN frame [790, 5, 33, 104, 9, 33, 34, 0, 111] and a spoofed frame [790, 69, 41, 36, 255, 41, 36,0, 255] differ across several payload values, resulting in more observable deviations from normal traffic that the model can exploit during learning. In addition, the model was trained and evaluated using the same hyperparameters as those employed for the dataset in Table 5.11, without dataset specific tuning in order to maintain experimental consistency. While hyperparameter optimisation could improve performance, it would require additional tuning effort for each manufacturer's dataset. Furthermore, the dataset used in Table 5.14 contains substantially fewer training samples, limiting the model's ability to learn representative patterns. As shown in Table 5.14, increasing the amount of normal training data leads to improved performance and a reduced FAR, enabling the model to better distinguish between normal and attack traffic.

Table 5.14 Detection results for unseen attacks for Car Hacking: Attack & Defence Challenge 2020 dataset [2].

Unseen attack	**Number of normal training data**	**F1**	**Pre**	**Acc (%)**	**DR (%)**	**FAR (%)**
Flooding	30,548	0.68	0.52	94.40	1.0	0.059
Frame Fuzzification	30,548	0.53	0.36	94.24	99.71	0.059
Spoofing	30,548	0.070	0.039	0.94	28.81	0.059
All	30,548	0.75	0.63	94.07	94.19	0.059
All	76,288	0.76	0.63	94.17	94.19	0.0583
All	190,597	0.78	0.67	94.90	94.20	0.0502

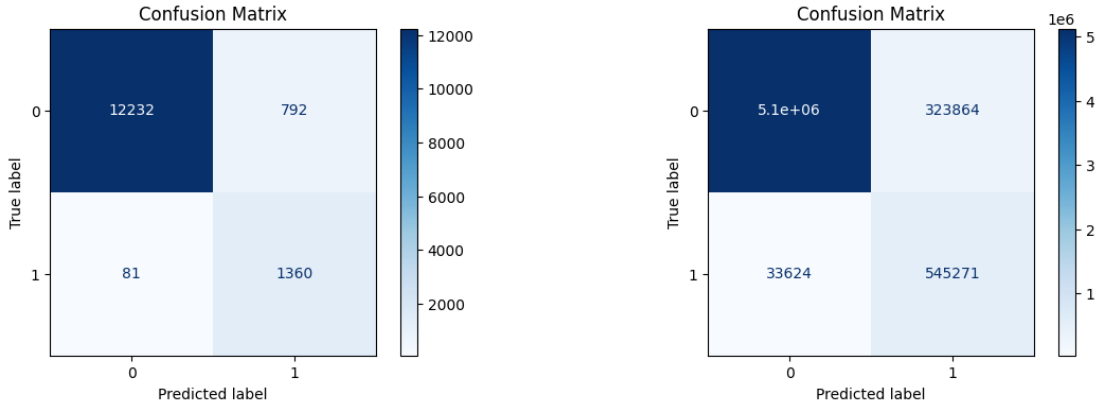
Moreover, Figure 5.11a shows the confusion matrix on the test samples after applying SMOT sampling to the second dataset, and Figure 5.11b displays the confusion matrix for the entire dataset.



(a) SMOT sampling.

(b) All remaining testing data.

Fig. 5.10 LSTM-autoencoder binary confusion matrices for Car Hacking dataset [1].



(a) SMOT sampling.

(b) All remaining testing data.

Fig. 5.11 LSTM-autoencoder binary confusion matrices for Car Hacking: Attack & Defence Challenge 2020 dataset [2].

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 [28]. As shown in Table 5.11, 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.

Adversarial Testing and Training:

In this section, we present the results on the performance of the proposed IDS when evaluated with adversarial samples. Figure 5.15 compares the results before and after the introduction of adversarial samples. The figure shows that the adversarial samples successfully bypass the model by manipulating input data, causing it to misclassify malicious inputs as normal and, in some cases, normal frames as malicious. As a result, in both datasets the F1-score decreased from 0.99 to 0.95 and the DR from 99.99% to 96.04% in the Car Hacking Dataset [1]. A similar trend was observed in the Car Hacking: Attack & Defence Challenge 2020 Dataset [2], where the F1-score decreased from 0.99 to 0.97 and the DR from 99.76% to 95.59%. Although adversarial performance degradation can differ between datasets, none of the commonly cited factors such as the number of samples, dimensionality, distribution type, density, concentration, class separation, label quality, or domain-specific properties [195] were applicable in our case. Therefore, the effect of the adversarial samples was similar in both datasets, resulting in a moderate and comparable reduction in performance.

Table 5.15 Performance evaluation of adversarial attacks

	Original samples	**Adversarial samples**	**Acc (%)**	**F1**	**Pre**	**DR(%)**
Car Hacking Dataset [1]						
Without adversarial samples	16569475	0	99.99	0.99	0.99	99.99
With adversarial samples	16569475	16569475	96.04	0.95	0.947	96.04
Adversarial training	16569475	16569475	99.98	0.99	0.99	99.98
Car Hacking: Attack & Defence Challenge 2020 Dataset [2]						
Without adversarial samples	6024814	0	99.77	0.99	0.99	99.76
With adversarial samples	6024814	6024814	95.59	0.97	0.99	95.59
Adversarial training	6024814	6024814	99.76	0.99	0.99	99.76

Adversarial training is one of the most effective approaches for defending ML- and DL-based models against adversarial examples, aiming to enhance model robustness intrinsically

[196]. To achieve this, adversarial samples are added to the training set, which consists of a mixture of adversarial and clean examples. As shown in Figure 5.15, the F1-score increased to 0.99 in both datasets when adversarial samples were included in the training set. Other metrics also improved, reaching values close to those observed before the introduction of adversarial samples. Training the IDS with diverse adversarial examples enhances the model's robustness against new unknown attacks by incorporating them into the learning process.

5.4.3 Model Complexity

This section discusses model complexity in terms of model size (in megabytes, MB), trainable parameters, and inference time and latency (in milliseconds, ms). When designing in-vehicle IDS solutions, it is essential to consider the deployment requirements [21]. 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 [27].

Regarding model size and trainable parameters, we have simplified the model architecture to minimise its size in pursuit of optimal results. 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 regularisation technique in the LSTM-autoencoder. Through careful experimentation with hyperparameters, we optimised 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 and have a smaller memory footprint [63], implying greater efficiency in energy consumption [30]. The ANN model has 517 trainable parameters, while the LSTM-autoencoder has 253,065, making a combined total of 253,582 trainable parameters.

Moreover, since CAN is a time-critical system, inference time and detection latency are essential safety-related metrics for in-vehicle IDS to ensure real-time performance. Inference time refers to the amount of time required for a trained model to generate predictions on a new data batch [94]. Latency, on the other hand, is the time taken for a packet to travel from its source to its destination [156]. The United States (US) Department of Transportation states that critical vehicle safety services, such as collision and attack warnings, should operate with a latency of 10 to 100 ms [197]. Meanwhile, Vehicle-to-Everything (V2X)-based autonomous and cooperative driving applications require even stricter latency, typically between 10 and

Table 5.16 Inference time and latency of the proposed IDS.

IDS Component	Data Pre-processing (ms)	Inference Time (ms)	Latency (ms)
ANN model	0.0060	0.0018	0.0078
LSTM model	0.0034	0.028	0.0314
Total			**0.0392**

20 ms [198]. Thus, for a vehicle-level IDS, the time required to process each network packet is required to be less than 10 ms to meet real-time requirements. We measure inference time using batches with a batch size of 256, as using a smaller batch size can reduce the overall detection latency [63]. Table 5.16 shows the inference time and latency per sample in both models. The total latency for a packet to pass through our proposed IDS is around 0.0392 ms, which is much lower than the required latency of 10 ms. This demonstrates the applicability of our IDS for real-time requirements.

5.4.4 Comparison with Existing Studies

This model is compared with recent work in [156], since they used the same dataset and a similar approach. 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 number of testing instances for attack and normal instances as were used in [156]. Results in Table 5.17 show that our model outperformed the results in [156], 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 [156] 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 [63] and [1] 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.

In summary, our analysis revealed several key findings that contribute to the understanding and development of in-vehicle IDSs, thereby advancing the field of vehicular security.

- Hybrid IDSs, such as our proposed IDS, can be a robust solution that not only addresses current attacks but also prepares for future ones. Moreover, the order of each approach is important. For example, we adopted the seen attack detection before anomaly

Table 5.17 Comparison with existing Work.

Unseen Attack	Ours			MTHIDS [156]		
	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

detection 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 [164, 165].
- The most important finding is that our results prove that DL algorithms can improve the performance of an IDS while meeting the model size and real-time requirements in resource-constrained environments.

Among the extensive body of previous work, only a few studies have considered measuring model size [156, 63, 158, 161, 160]. Accordingly, we compared our model with theirs. As shown in Table 5.18, our trainable parameters represent approximately an 88.2% reduction compared to the number of trainable parameters in [63], despite their use of only one feature (the CAN ID), while we utilised all the CAN frame features. Additionally, our model size is smaller than that of the IDS proposed in [158].

Furthermore, the model size in [156] for the two models is 2.61 MB, while the total size of our models is 2.98 MB, demonstrating that our model remains within a similar range, even though we used DL, which is typically more resource-intensive than traditional ML. These sizes are well below the typical memory capacity of vehicle-level machines, which can exceed 1 GB of RAM [156]. Moreover, recent studies [161] and [160] have also demonstrated reductions in model size.

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.

Table 5.18 Model size comparison.

Model	Model size	Trainable parameters
MTHIDS [156]	2.61 MB	-
AE- GAN [63]	-	2.15 million
BNN, GAN[158]	4.07 MB	-
VAE- AERL[161]	2,542 KB	-
Deep CAN IDS [160]	0.01 - 0.3MB	-
Ours	2.98 MB	253,582

5.5 Conclusions

The aim of this chapter was to propose a robust and lightweight multi-stage IDS designed for in-vehicle network security that is capable of detecting both seen and unseen attacks. Our IDS addresses the limitations of existing solutions by utilising a hybrid approach and advanced DL algorithms. We evaluated the performance of our IDS using two real-world datasets containing various cyberattacks, including DoS, frame fuzzification, and various types of 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 unknown attacks, achieving an F1-score of over 0.95 and a DR of 99.99% with minimal false alarms. It also maintains a compact model size of 2.98 MB and a latency of less than 0.04 ms, meeting the real-time requirements of vehicle safety services. Overall, our proposed IDS detects both seen and unseen 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 unseen attacks, our IDS not only addresses current attacks but also prepares for future ones. However, despite significant advancements in existing in-vehicle IDSs, challenges related to data privacy and communication efficiency persist in centralised environments, where data is sent to a cloud server for training. The FL approach offers a promising solution by enabling local model training while preserving the privacy of raw data. In the next chapter, we will explore the field of FL-based IDSs in in-vehicle networks, simulate the deployment of our proposed IDS, and evaluate its performance in a simulated hierarchical FL environment.

Chapter 6

Hierarchical Federated Learning-based Intrusion Detection for In-Vehicle Networks [1]

In this chapter, we aim to address the limitations of standard FL-based IDSs—which rely on a single central aggregator, leading to performance bottlenecks and introducing a single point of failure that compromises robustness and scalability—by proposing a Hierarchical Federated Learning (H-FL) framework. The H-FL framework integrates multiple edge servers (aggregators) alongside the central aggregator, mitigating single-point failure risks, enhancing scalability, and efficiently distributing the computational load. Experimental results demonstrate that deploying the IDS within the H-FL framework can improve the F1-score by up to 10.63%, addressing the limitations of edge-FL in terms of dataset diversity and attack coverage. Notably, H-FL achieved improved F1-scores in 16 out of 24 evaluated scenarios.

6.1 Introduction

As highlighted in previous chapters, ML and DL-based IDSs have proven effective in detecting attacks within in-vehicle networks. However, their reliance on centralised training, which involves transmitting raw data to a central server for learning or storage [76], raises concerns about data privacy and limits communication efficiency [76]. To address these issues, FL offers a promising solution by enabling local model training while preserving the

[1] This chapter is published in *Future Internet* Journal 16.12 (2024): 451.

privacy of raw data [34]. FL addresses key limitations of centralised IDS approaches and is particularly well suited for in-vehicle networks for several compelling reasons:

- **Privacy preservation:** Regulations such as the General Data Protection Regulation (GDPR) [77], California Consumer Privacy Act (CCPA) [78], Personal Information Protection and Electronic Documents Act (PIPEDA) [79], and Brazilian General Data Protection Law (LGPD) [80] safeguard sensitive data from unauthorised movement. FL aligns with these regulations by transmitting only the learned parameters to the central server, rather than raw data, thereby enhancing data privacy and protection [70].
- **Reduced latency**: By training data locally and transmitting only model updates, FL significantly reduces latency compared to traditional centralised approaches [76, 73], which require sending raw data to a central server [70].
- **Compliance with guidelines**: According to the 2020 guidelines of the International Telecommunication Union, an in-vehicle IDS must have the ability to regularly update its set of rules [81]. FL supports this requirement by enabling continuous learning and iterative model refinement.
- **Diverse driving scenarios**: FL facilitates the training of a universal model that encompasses various driving scenarios, vehicle states, and driving behaviours [38].

This chapter aims to answer the following research questions:

RQ5: *How can in-vehicle IDSs detect new attacks that were never part of the central server's coverage in the FL architecture?*

RQ6: *How can the architecture of FL-based in-vehicle IDSs be enhanced to mitigate network congestion and communication delays?*

RQ7: *How can we reduce the risk of single-point failure inherent in centralised FL architectures?*

In answering these questions, the following contribution is made:

C5 To propose and deploy our novel in-vehicle IDS within a simulated H-FL framework for interconnected and autonomous vehicles, incorporating multiple edge servers (aggre-

gators) alongside a central aggregator, while accounting for realistic non-independent and identically distributed (non-IID) data to enhance the representation of real-world variability in FL environments.

As discussed in Chapter 3, numerous studies have developed FL-based IDSs for in-vehicle networks [35–39, 169, 171, 145]. However, they all utilise the standard cloud-based FL architecture. To the best of our knowledge, this is the first study to introduce the H-FL framework for an in-vehicle IDS and to simulate the deployment and evaluate IDS performance within this architecture. The aim of this chapter is to present and evaluate the deployment of the proposed IDS from Chapter 5 within the simulated H-FL environment.

6.2 Background and Related Work

In Chapter 3, we reviewed related work on FL-based IDSs. In this section, however, we provide background information on cloud-based FL and H-FL, examine the data distributions considered in these studies, and discuss the limitations of existing research.

6.2.1 Cloud-Based FL and Hierarchical FL

The standard cloud-based FL architecture comprises a central cloud server and numerous clients, as shown in Figure 6.1. Clients (vehicles) download a global model from the central server, perform multiple rounds of local training, and then send the updated model weights back to the central server for aggregation. This process is repeated until the model reaches the desired level of accuracy. In cloud-based FL, the number of participating clients can reach into the millions [40], which may lead to slow and unpredictable communication with the central server due to factors such as network congestion, ultimately resulting in inefficiencies in the training process [40, 199]. Furthermore, reliance on a single aggregator introduces a single point of failure [41], which can compromise the availability of the entire FL framework. In contrast, the H-FL architecture comprises a central server, multiple edge servers, and numerous clients. In H-FL, clients update their local parameters and send them to the nearest edge server for aggregation, following a similar approach to cloud-based FL. The key difference is that, after several rounds of edge-level aggregation, the edge servers transmit their aggregated parameters to the central server for global aggregation. This hierarchical design enables broader client participation and significantly reduces costly communication with the central server by leveraging more efficient client–edge interactions, resulting in substantial reductions in both runtime and the number of required local iterations [47].

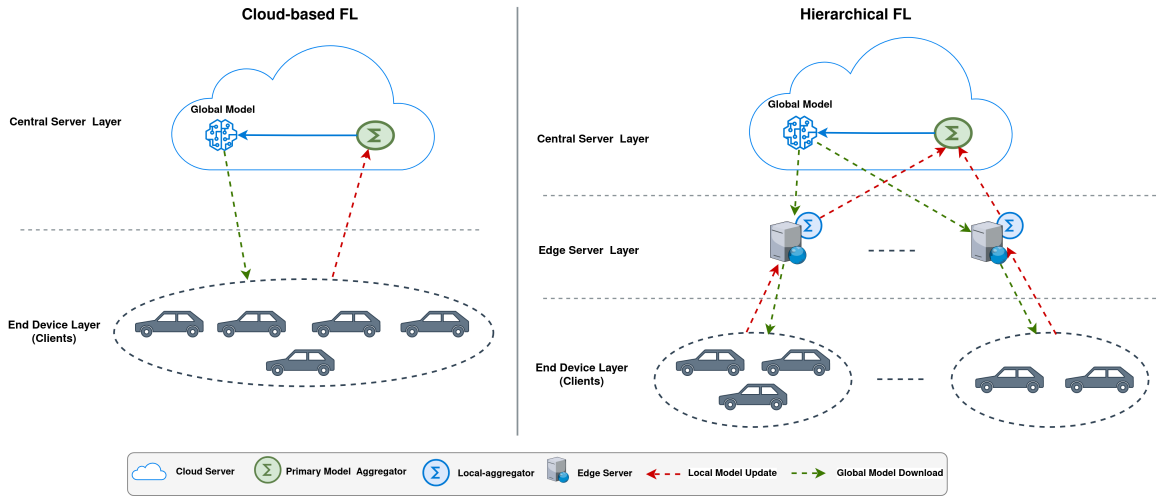


Fig. 6.1 Cloud-based FL and H-FL.

6.2.2 Non-Independent and Identically Distributed Data Distributions

Data distribution among clients is a critical factor in the FL process. In real-world applications, training data are rarely independent and identically distributed (IID); instead, they are typically non-IID due to variations in user behaviour, preferences, and environments. Non-IID data arise when the local training data differ significantly across clients, resulting in heterogeneous data distributions [199]. However, previous works [37, 36, 38, 35, 171, 145] did not account for non-IID data, instead employing data partitions in which clients received either an equal number of samples or samples from all classes (i.e., types of attacks). This approach contradicts the nature of typical FL scenarios, which are characterised by non-IID data distributions [70], and relies on an unrealistic assumption about data homogeneity across clients [200]. Non-IID data distributions are only considered in [39, 169]. In [39], nine candidate clients are assumed, each possessing varying numbers of data samples (50, 100, 150, 1000, 1500, 2000, 2500, 3000, 3500), but the data distribution among clients and the distribution of samples across all classes are not clarified. In [169], to achieve a non-IID setting, data are allocated to vehicles using a Dirichlet (μ) distribution technique, with the (μ) parameter adjusted from 0.1 to 0.7. However, as shown in Table 6.1, most existing studies [35–37] did not take into account the inherent data heterogeneity (non-IID characteristics) of CAN bus data when partitioning it across clients, which could affect the applicability of their findings to real-world FL scenarios [200]. This consideration is essential for two reasons: first, in real-world deployments, vehicles will not have uniform distributions of both attack and normal data; second, testing with varying non-IID levels enables a more comprehensive evaluation of IDS performance and robustness across diverse data distributions.

Table 6.1 FL-based IDSs for in-vehicle network

Reference	FL	Non-IID	Aggregation Function	Dataset	FL Implementation
[169]	Standard	✓	FedAvg	car-hacking [85]	PyTorch
[37]	Standard	x	FedAvg	HCRL CAN Intrusion Detection [61]	N / A
[36]	Standard	x	FedAvg	car-hacking [85], NAIST CAN attack dataset[66]	Keras, TensorFlow
[38]	Standard	N / A	FedAvg, FedProx	READ [170]	N / A
[35]	Standard	x	FedAvg	Car Hacking: Attack & Defence Challenge 2020 [2]	Keras, TensorFlow
[39]	Standard	✓	FedAvg	HCRL CAN Intrusion Detection [61]	N / A
[145]	Standard	N / A	FedAvg, FedProx	Recan [154]	N / A
[171]	Standard	-	FedAvg	Own	Pytorch, Flower
Our work [44]	Hierarchical	✓	FedAvg	car-hacking [85], Car Hacking [1]	Flower

To address the limitations of previous studies, namely their reliance on standard cloud-based FL architectures and limited consideration of realistic data distributions. Our proposed H-FL framework employs a central aggregator in conjunction with multiple edge servers (aggregators), which offers several advantages, including reduced dependence on a single central server. Furthermore, both experimental results and theoretical analyses in other domains have shown that the H-FL architecture leads to faster convergence, lower training time, and reduced energy consumption on end devices compared to the standard cloud-based FL framework [47]. To the best of our knowledge, no prior work has deployed and evaluated the performance of an in-vehicle IDS in an H-FL environment.

6.3 Materials and Methods

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

6.3.1 Experimental Setup

The implementation was conducted in Visual Studio Code. The experimental setup used a 64-bit Windows 11 Pro for Workstations operating system with an AMD Ryzen Threadripper PRO 5995WX processor, featuring 64 cores at 2701 MHz, 128 logical processors, and 128 GB of RAM. The construction, training, and evaluation of the DL model were conducted with Flower framework [201], TensorFlow (version 2.13.1), and Python (version 3.8.10).

Flower is an open-source FL framework to build AI applications capable of training on data distributed across numerous devices. We selected Flower for its ability to seamlessly transition from conventional to federated ML frameworks, its compatibility with popular platforms such as TensorFlow and PyTorch, its support for diverse privacy requirements, and its flexibility in facilitating the implementation of novel approaches with minimal engineering effort [202].

6.3.2 Hierarchical Federated Learning Architecture

The H-FL architecture consists of a central server, several edge servers (aggregators), and numerous clients (vehicles), as depicted in Figure 6.2. The central server is connected to the edge servers, each of which interacts with vehicles in different geographical locations. In this study, we utilise one central server, two edge servers, and five clients per edge. The H-FL process begins with the central server sending the initial model to the edge servers. Each edge server then selects a subset of clients for training using a suitable selection approach. The selected clients receive the initial model and perform edge aggregations over several rounds to generate an edge model. Once the edge servers complete their aggregations, they send their aggregated models' parameters to the central server. The central server aggregates these edge models to produce a global model, which is then sent back to the edge servers for further aggregation. A detailed description of the H-FL process is provided in Algorithm 3. To handle the variability in CAN bus data across different vehicle makes and models, edge servers select clients based on similarities in their CAN bus data, such as make or model, within specific geographical areas. This approach ensures that the global model is trained on relevant and similar data, thereby improving accuracy.

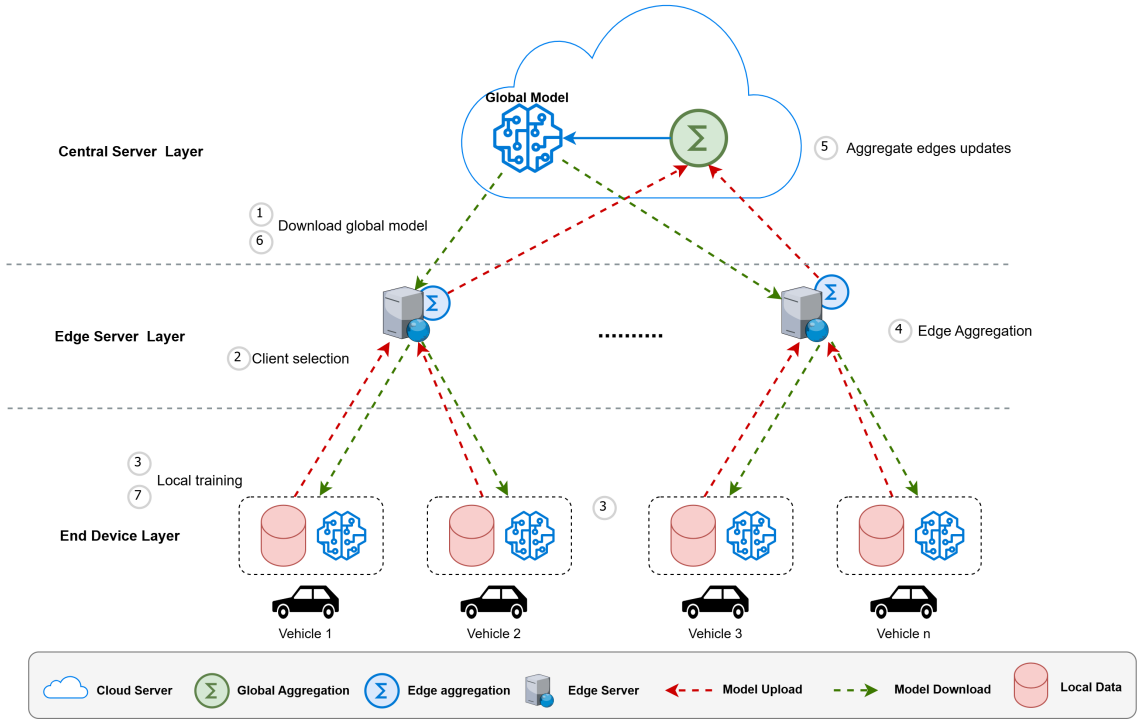


Fig. 6.2 Architecture of the proposed H-FL method.

Algorithm 3 Detailed description of the H-FL approach.

Begin

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

End

Figure 6.3 illustrates this H-FL setup as implemented in Flower. The central server, located at 127.0.0.1:8088, initiates the process by distributing the pre-trained model to its connected clients (edge servers). Subsequently, each edge, serving as a server at addresses

127.0.0.1:8080 and 127.0.0.1:8081, respectively, relays the pre-trained model to its connected clients (vehicles). Each edge server then starts the standard FL process with the clients (vehicles) in its area, resulting in an edge model. These edge servers, now operating as clients, send their models back to the central server. The central server aggregates the received edge models into a global model, which it then redistributes to the edges. Finally, the edges, again functioning as servers, transmit the new global model to their clients (vehicles) to restart the FL cycle.

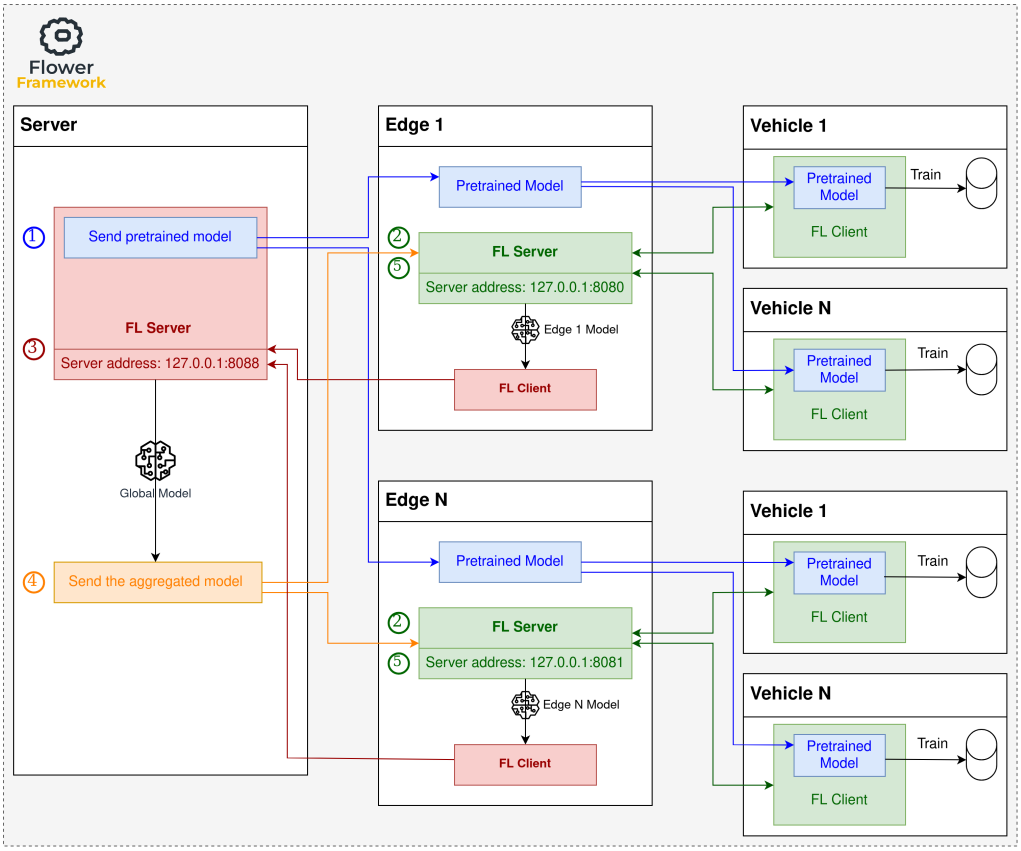


Fig. 6.3 H-FL environment in Flower.

6.3.3 Dataset Description

To evaluate the performance of our IDS within the federated framework, we employed a benchmark car hacking dataset published by Song et al. [1]. We selected this dataset due to its widespread use in automotive security research and its status as the most frequently cited resource in the literature of the field [27]. The dataset contains normal data and four types of attacks: Denial of Service (DoS), frame fuzzification, engine RPM spoofing, and drive gear

spoofing. It includes four files, one for each attack (DoS, frame fuzzification, gear spoofing, and RPM spoofing), with each file including normal as well as attack instances. Table 6.2 shows the number of instances for each type of attack as well as for normal data. For every CAN message, the dataset provides valuable information, including timestamp, CAN ID, DLC, data field, and flag. The timestamp records the exact time the message was captured since the system startup, while the CAN ID determines message priority, with lower values being prioritised. Additionally, the DLC defines the length of the data field, which can reach a maximum of 8 bytes. The flag indicates whether the message is normal or an attack. Table 6.3 provides an overview of the dataset's features, along with their descriptions and data types. To ensure the generalisability of our proposed H-FL architecture across diverse scenarios and to avoid reliance on a single dataset, we validated it using another benchmark dataset in automotive security research, the car hacking: attack & Defence Challenge 2020 dataset [2], which comprises normal traffic and three types of attacks: DoS, frame fuzzification, and spoofing [2].

Table 6.2 Dataset overview.

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

Table 6.3 Data features, descriptions, and types.

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

6.3.4 Data Pre-processing and Partitioning

To replicate a real-world scenario where data are independently generated for each client, each FL client performs individual and independent data pre-processing before engaging in local model training, as depicted in Figure 6.4. However, certain pre-processing steps, such as removing missing data and label encoding, should occur before data partitioning. This approach ensures that each client receives a consistent and complete dataset, fostering a standardised training environment and maintaining uniform label representation throughout the FL process. Therefore, we divide the data pre-processing into two phases: initial data pre-processing and on-device data pre-processing. Initial data pre-processing occurs before data partitioning, while on-device pre-processing is performed on the device. This section will elaborate on each step: initial data pre-processing, data partitioning, and on-device data pre-processing.

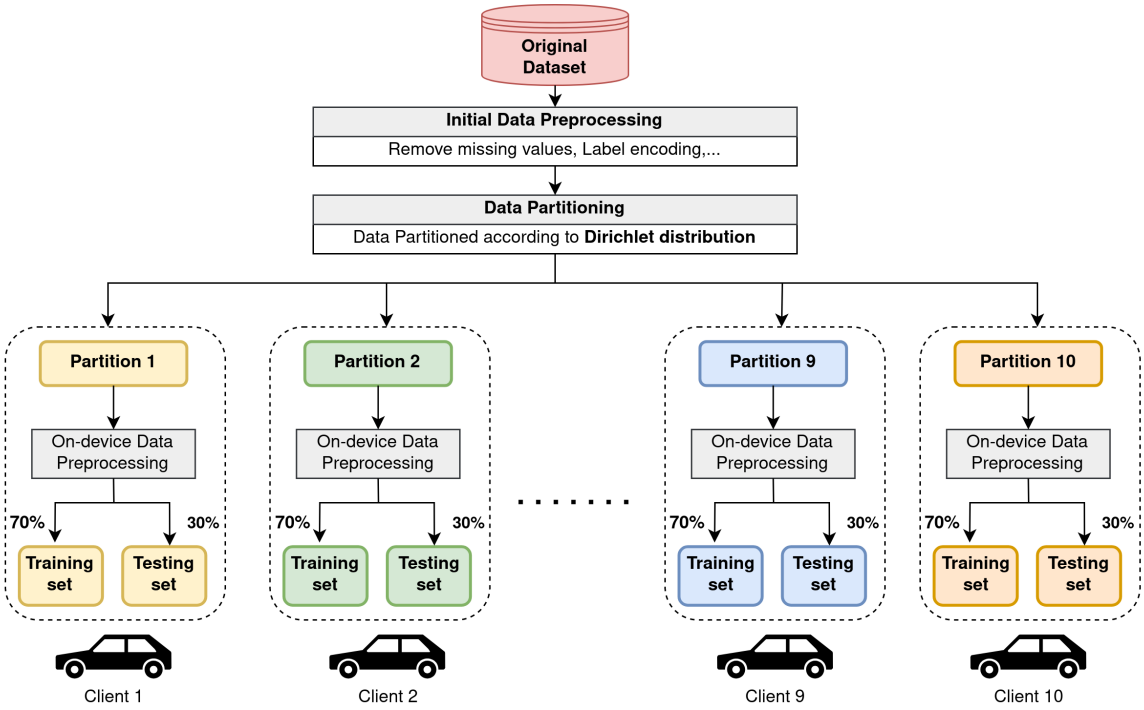


Fig. 6.4 Data partitioning.

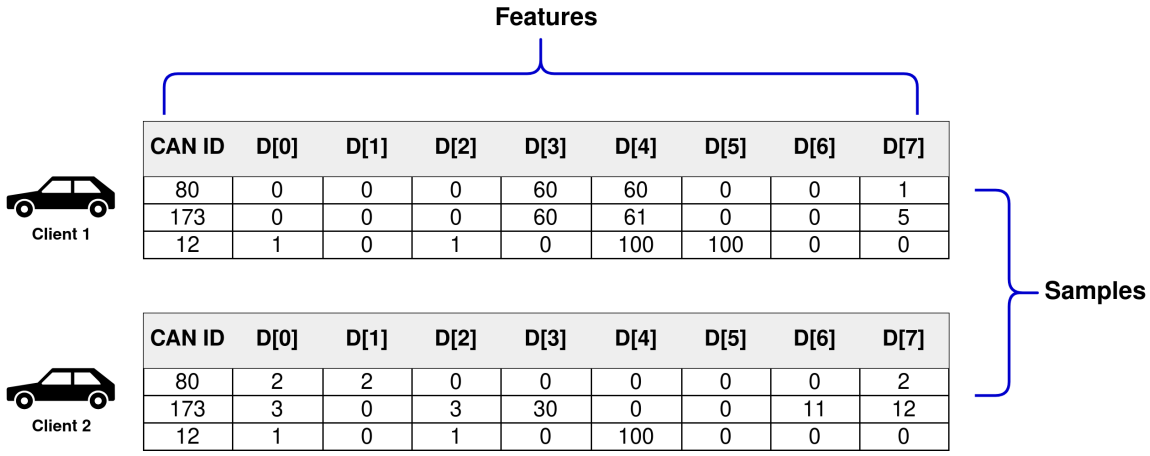
Initial Data Pre-processing

In the initial phase of data pre-processing, for each data file, we convert 'T' to the corresponding attack type, such as 'DoS', and 'R' to 'Normal' in the 'Flag' column. Then, we shift the 'Flag' field to the last column and fill unavailable data bytes with (NaN) values. We

combine the four data files into one. Given the extensive data points we have, we remove any row with these missing values (NaN) in the Data fields. Lastly, we use a label encoder to convert categorical values into numerical representations. This ensures consistent encoding across all clients, facilitating model training and aggregation.

Data Partitioning

After the initial data pre-processing step, we perform data partitioning to simulate the FL environment. In the context of CAN bus data, the data are horizontally structured, where the training data from participating clients shares a common feature space but exhibits distinct sample spaces. As illustrated in Figure 6.5, for instance, client 1 and client 2 contain dissimilar data while sharing the same set of features.



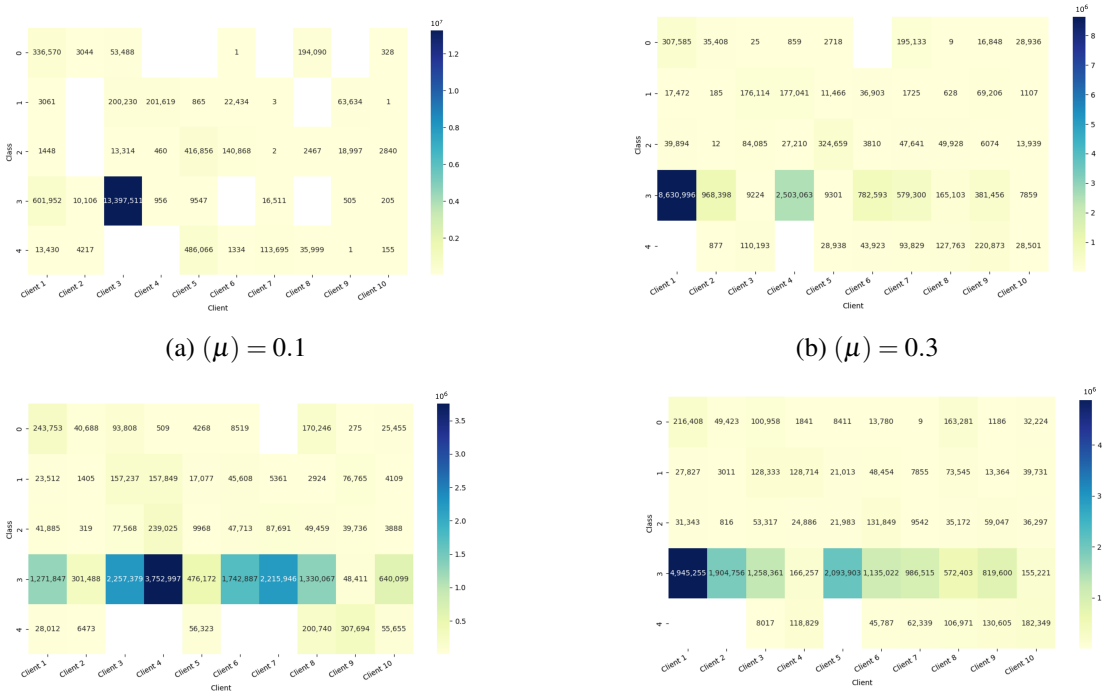
CAN ID	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]
80	0	0	0	60	60	0	0	1
173	0	0	0	60	61	0	0	5
12	1	0	1	0	100	100	0	0

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

Fig. 6.5 Horizontal CAN bus data.

To simulate a non-IID setting, we assign data to clients using Dirichlet $(Dir(\mu))$ distribution according to [203], where every client is assigned a share of the samples for each label. Dirichlet distribution is commonly used as a prior distribution in Bayesian statistics [204] and is an appropriate choice to simulate real-world data distribution [203]. An advantage of this approach is that we can flexibly change the imbalance level by varying the concentration parameter (μ). The parameter (μ) in $(Dir(\mu))$ regulates the degree of non-IID property among clients, where lower (μ) values signify a higher level of non-IID. We adjust (μ) to simulate diverse data distributions among clients, ranging from 0.1 (indicating the highest non-IID) to 0.7 (indicating the lowest non-IID). Figure 6.6 and Figure 6.7 show the data partitioning for the Car Hacking dataset [1] and the Car Hacking: Attack and Defence Challenge 2020 dataset [2] under different (μ) values. The number displayed in each rectangle represents

the quantity of data samples of a class (DoS (0), frame fuzzification (1), drive gear spoofing (2), normal (3), and engine RPM spoofing (4)) for each client. However, we exclude (μ) = 0.1 due to its highly imbalanced data distribution, as shown in Figure 6.6a, where some clients lack normal data (3) for training, making it unrealistic. Data are distributed among two edges, each with five clients, for a total of ten clients. We chose ten clients, a common practice in the literature [38, 205–207], to manage the increased computational demands associated with additional clients.



(a) $(\mu) = 0.1$

(b) $(\mu) = 0.3$

(c) $(\mu) = 0.5$

(d) $(\mu) = 0.7$

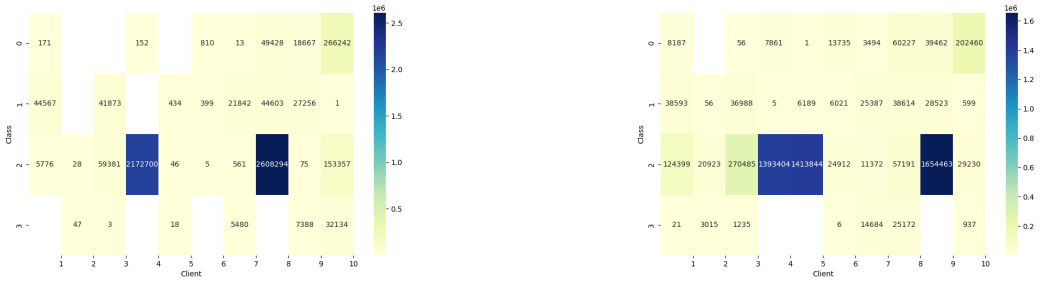
Fig. 6.6 Car Hacking dataset [1] distribution based on Dirichlet(μ) distribution.

On-Device Data Pre-processing

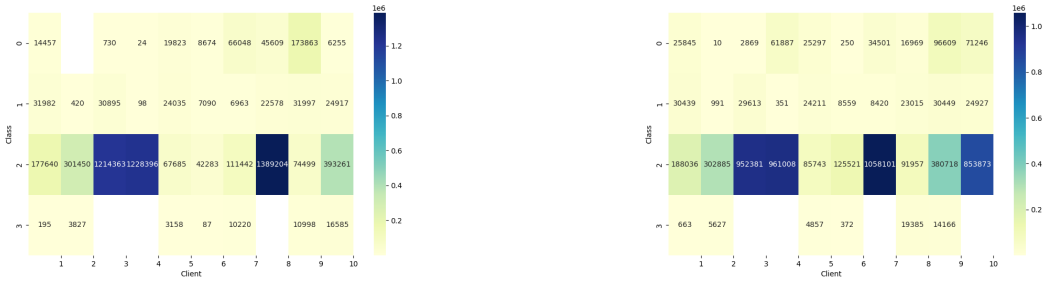
For on-device data pre-processing, unwanted columns (Timestamp, DLC) are removed. Additionally, we convert the CAN ID and Data values from hexadecimal to decimal values per the ML requirement, and a normalisation process is applied.

6.3.5 Model Initialisation and Pre-Training

The model is initialised on the server side, followed by distributing a copy to each participating client. In this chapter, we use a pre-trained model, as Chen et al. [208] demonstrate



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



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

Fig. 6.7 Car Hacking: Attack and Defence Challenge 2020 dataset [2] distribution based on Dirichlet(μ) distribution.

that pre-training models on a central server before distribution to edge devices enhances task accuracy, especially with non-IID client data. Additionally, many advantages arise from pre-training a model on a centralised dataset for FL. Firstly, it enhances generalisation by enabling the model to capture general features and patterns, ensuring better adaptability to the diverse data distributions across devices. Furthermore, initialising a pre-trained model on a powerful centralised server can significantly reduce the time required to fine-tune the model for specific devices [209]. To pre-train the models, we extract a small data sample representing each class, amounting to 0.8% (less than 1%) of the entire original dataset, for use in pre-training. The remaining data are then partitioned across clients. Our approach builds on the methodology outlined in [156], employing K-Means clustering to capture underlying patterns and the SMOTE method to address class imbalance. The sampling process starts by grouping the data into clusters using K-Means. To ensure a representative subset, we apply a group-based sampling strategy, selecting 0.8% of data points from each cluster. This approach retains diversity in the sampled dataset while significantly reducing its size. The selected samples are subsequently excluded from the full dataset before partitioning

the data among clients. As the sampled data often exhibit label imbalance, SMOTE is applied to balance the classes.

6.3.6 Local Training

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

6.3.7 Aggregation

In H-FL, there are two stages of aggregation: edge aggregation and server aggregation. Edge aggregation combines the models from the clients, while server aggregation merges the results from edge aggregation to produce a global model. An aggregation strategy refers to the process of combining local models from clients into a centralised global model. In this chapter, we use the standard aggregation strategy for both edge server aggregations, namely Federated Averaging (FedAvg) [199], according to the following equation:

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

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

6.3.8 Evaluation

In FL, there are two ways to evaluate the model: on the server side or the client side. In this chapter, we chose to employ client-side evaluation over server-side evaluation because it enables us to assess models across a larger set of data, often leading to more realistic evaluation outcomes [210].

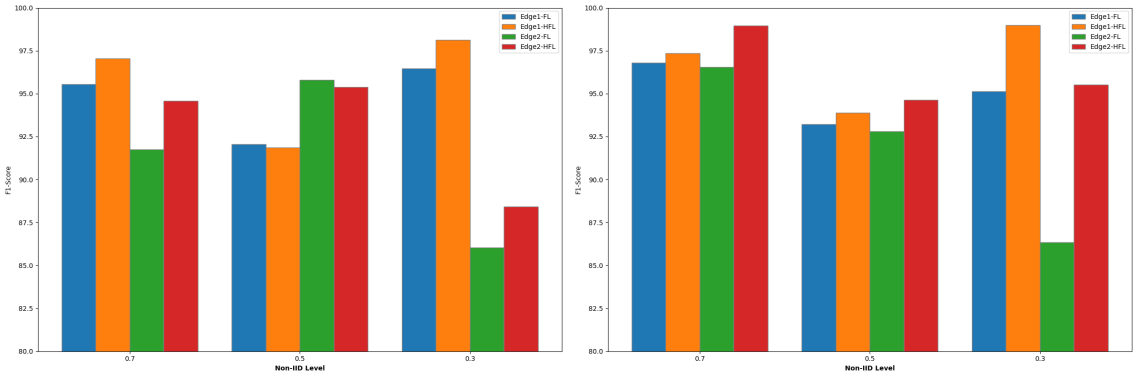
6.4 Results and Evaluations

6.4.1 Evaluation Metrics

For a robust assessment of model performance, it is important to choose metrics that align with the dataset type and class distribution. In the context of imbalanced intrusion datasets, relying solely on accuracy is considered unreliable [211]. Hence, we opted for the evaluation metrics of F1-score, precision, and recall for our model. Furthermore, given the unequal distribution of the data based on the label, we use the weighted average to consider the size of each class. This is an important consideration, as it ensures that the distribution of instances in each class is taken into account [212]. Additionally, the training time for each model, network load, and memory requirements for each client are also evaluated.

6.4.2 Performance Results and Analysis

We evaluate the performance of the proposed IDS within an H-FL architecture using two datasets, each with three non-IID levels, resulting in a total of 24 scenarios: 12 for dataset 1 and 12 for dataset 2. Figure 6.8 shows the average F1-score performance of the ANN model and the proposed cascaded IDS across different non-IID levels in the car hacking dataset [1]. As depicted in Figure 6.8a, the ANN classifier's performance improved at all non-IID levels except when $(\mu = 0.5)$, where it slightly decreased after applying the H-FL model. In contrast, the F1-score for the cascaded multistage IDS increased across all non-IID levels, as shown in Figure 6.8b. Notably, the H-FL improved the F1-score in 10 out of 12 scenarios in the car hacking dataset. Our results align with those of Liu et al.[47], who applied the H-FL framework in a different domain and on distinct data.



(a) ANN model.

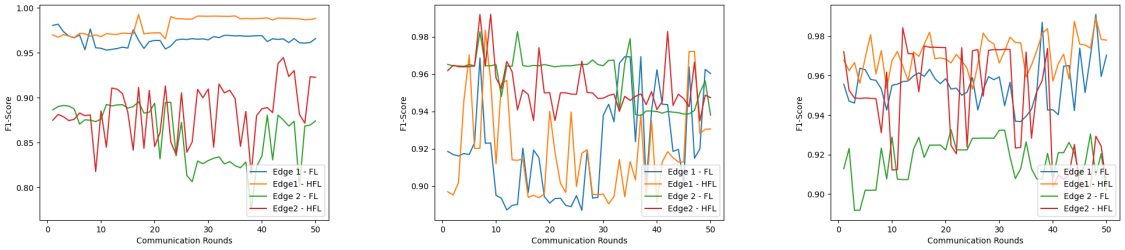
(b) Proposed multistage-IDS.

Fig. 6.8 Average F1-score results non-IID levels across communication rounds for car hacking dataset [1].

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

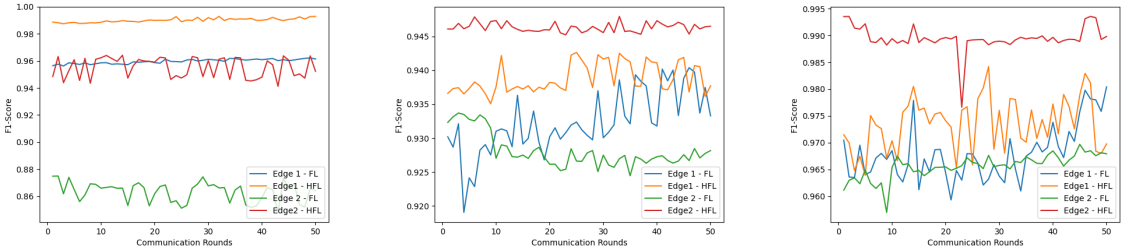
Figure 6.8b also reveals a notable difference between Edge 1 and Edge 2, primarily due to the high imbalance in data distribution when $(\mu = 0.3)$, compared to $(\mu = 0.5)$ and $(\mu = 0.7)$. Across all non-IID distributions, the H-FL approach consistently outperforms the edge-FL approach, with percentage increases ranging from 0.57% to 10.63%, highlighting a significant performance improvement with the H-FL method. The increase may vary based on the clients selected under each edge and non-IID level, as demonstrated by the Edge 2 results in Figure 6.8b. These results underscore the critical impact of the participating clients' data on performance. Detailed F1-score performance is available in Figures 6.9 and 6.10, which further illustrate the F1-score performance of the ANN model and the proposed cascaded IDS across various non-IID levels. Table 6.4 shows the average recall and precision

of our IDS in an H-FL environment. The results indicate an increase in both recall and precision after deploying the H-FL model at nearly all non-IID levels. The best results are obtained when the data are distributed across clients with a low level of imbalance, such as at $(\mu = 0.7)$. Figure 6.11 demonstrates the distributed loss across communication rounds. Results show that the model has different convergence speeds across non-IID levels.



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

Fig. 6.9 F1-score of ANN model for different non-IID levels across communication rounds.

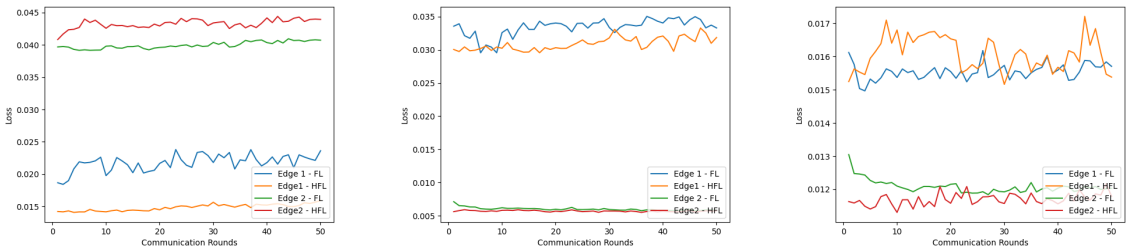


(a) $(\mu) = 0.3$ (b) $(\mu) = 0.5$ (c) $(\mu) = 0.7$

Fig. 6.10 F1-score of our proposed multistage IDS for different non-IID levels across communication rounds.

Table 6.4 Average recall and precision of the proposed IDS model.

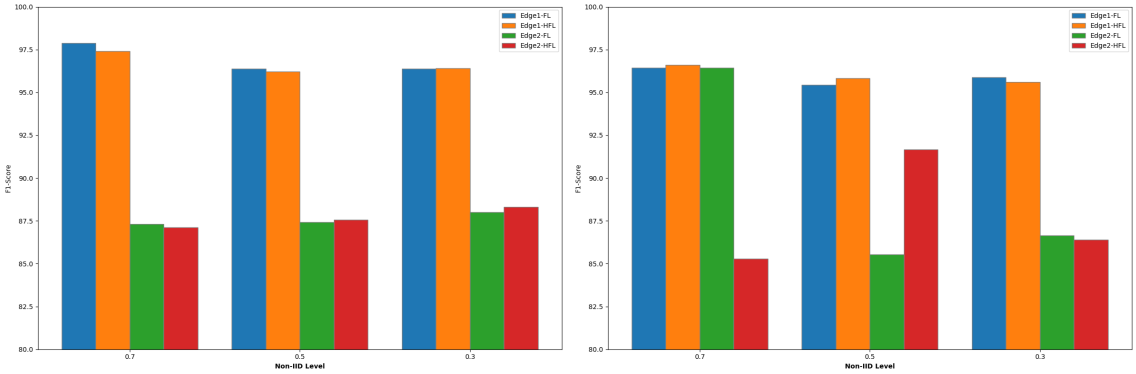
Non-IID	**Edge 1 FL**		**Edge 1 H-FL**		**Edge 2 FL**		**Edge 2 H-FL**	
	Recall	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**
0.3	94.93	96.76	98.39	99.89	81.29	84.61	86.48	94.71
0.5	92.59	95.27	93.91	94.21	92.92	93.11	95.40	94.16
0.7	95.28	98.63	95.75	99.53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



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

Fig. 6.11 Distributed loss across communication rounds.

We also evaluated the performance of the proposed IDS using another dataset, the car hacking: attack & Defence Challenge 2020 dataset. Figure 6.12 shows that 50% of the scenarios demonstrated improvement, while the remaining 50% showed a decline. This is likely due to imbalanced data distributions between labels and data heterogeneity. Overall, H-FL improved the F1-score in 16 out of 24 evaluated scenarios across both datasets.



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

Fig. 6.12 Average F1-score results non-IID levels across communication rounds for car hacking: Attack & Defence Challenge 2020 dataset [2].

Across the 24 evaluated scenarios, comprising 12 scenarios for each dataset, HFL leads to improved IDS performance in 16 cases, while the remaining 8 scenarios exhibit a slight degradation. The following analysis examines why HFL is effective in most scenarios and identifies the conditions under which its performance degrades. As shown in Table 6.5, which summarises HFL performance across scenarios and non-IID levels, the Car Hacking dataset [1] shows performance improvements under HFL across all non-IID settings (0.3, 0.5, and 0.7) and model types (ANN and ANN–LSTM IDS), except for the ANN multiclass model at the 0.5 non-IID level. Overall, 10 of the 12 evaluated scenarios in this dataset benefit from HFL. Although the data at this level are more balanced than in the non-IID 0.3 setting, class

imbalance persists at the client-level. As shown earlier in Section 6.3.4 in Figure 6.6, this non-IID configuration differs fundamentally from the other settings. Several clients (Clients 3, 4, 6, and 7) lack samples from five attack classes, leading to severely incomplete local class distributions. Consequently, local models are unable to learn meaningful decision boundaries for the missing classes, and the hierarchical aggregation process propagates these biased representations. This imbalance undermines the effectiveness of hierarchical aggregation for the ANN multiclass model and directly explains the observed performance degradation in this dataset. Therefore, in these scenarios, misclassification primarily stems from insufficient exposure to specific attack classes during local training, compounded by the aggregation of biased updates that distort global decision boundaries. In contrast to the first dataset, where HFL improves performance in the majority of scenarios, the Car Hacking: Attack and Defence Challenge 2020 dataset [2] shows improvements in only 6 of the 12 evaluated scenarios after applying HFL. This trend is consistent with the centralised IDS results for this dataset, which also demonstrate comparatively weaker performance. This behaviour can be explained by several factors. First, the relatively small number of data instances limits the models' ability to generalise effectively. Second, as observed in the first dataset, the absence of several attack classes across clients leads to incomplete local class distributions, preventing local models from learning robust decision boundaries. Finally, the attack traffic in this dataset closely resembles normal behaviour, making accurate detection challenging even in centralised settings. Under these conditions, hierarchical aggregation may induce negative transfer, as heterogeneous and biased local updates are combined, resulting in a slight degradation in overall performance. Overall, HFL performs well when clients retain sufficient class diversity and data volume to support meaningful local learning, but degrades in scenarios characterised by extreme class sparsity, limited samples, and high similarity between attack and normal traffic.

Table 6.5 Summary of HFL performance across scenarios and non-IID levels

Dataset	Scenario	Non-IID level	Model	Edge	Improved	Analysis
152	1	0.3	ANN	Edge 1	yes	In this scenario, the ANN multiclass classification model learns to classify the data more effectively after applying HFL, despite the severe class imbalance between clients.
	2		ANN	Edge 2	yes	Similar results are observed for Edge2, where the ANN multiclass classification model learns to classify the data more effectively after applying HFL, despite the severe class imbalance between clients.
	3	0.5	ANN	Edge 1	No	In this scenario, the data are more balanced than in the previous non-IID level (0.3); however, the ANN multiclass classification model shows a slight decrease in classification performance after applying HFL.
	4		ANN	Edge 2	No	Similar results are observed for Edge 2, where the ANN multiclass model shows a slight performance decrease after applying HFL.
	5	0.7	ANN	Edge 1	yes	In this scenario, the data are more balanced, and the ANN model benefits from HFL, resulting in improved classification performance.
	6		ANN	Edge 2	yes	Similar results are observed for Edge 2, where the data are more balanced and the ANN multiclass classification model benefits from HFL, leading to improved classification performance.
	7	0.3	ANN-LSTM IDS	Edge 1	yes	Despite the highly imbalanced data in this scenario, both the ANN model and the overall IDS benefit from HFL, achieving improved performance.
	8		ANN-LSTM IDS	Edge 2	yes	Similarly, in this scenario, the data are highly imbalanced, yet both the multiclassifier ANN and the overall IDS benefit from applying HFL, showing improved performance.
	9		ANN-LSTM IDS	Edge 1	yes	In this scenario, the data are less imbalanced; although the ANN shows a decline in performance, the overall IDS improves and benefits from applying HFL.

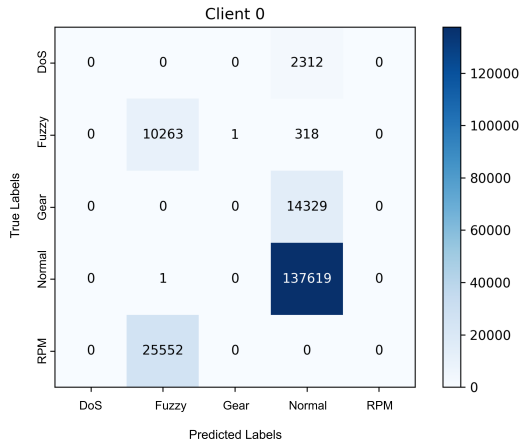
Continued on next page

Dataset	Scenario	Non-IID level	Model	Edge	Improved	Analysis
Car Hacking Dataset [1]	10	0.5	ANN-LSTM IDS	Edge 2	yes	Similarly, for Edge 2, the data are less imbalanced; although the ANN model shows a performance decline, the overall IDS improves under HFL.
	11	0.7	ANN-LSTM IDS	Edge 1	yes	In this scenario, the data are less imbalanced than in previous scenarios, and both the multiclassifier ANN and the overall IDS improve and benefit from applying HFL.
	12		ANN-LSTM IDS	Edge 2	yes	Similarly, for Edge 2, the data are less imbalanced than in earlier scenarios, and both the ANN multiclassifier and the overall IDS benefit from HFL, showing improved performance.
	13	0.3	ANN	Edge 1	yes	In this scenario, the performance of the ANN multiclass classification model shows a slight improvement in classifying the data more effectively after applying HFL, despite the severe class imbalance between clients.
	14		ANN	Edge 2	yes	Similarly, for Edge 2, the ANN multiclass classification model achieves a slight improvement in classification performance following the application of HFL, despite the severe class imbalance between clients.
	15	0.5	ANN	Edge 1	No	In this scenario, the data are more balanced than at the previous non-IID level (0.3); however, the ANN multiclass classification model exhibits a slight reduction in classification performance after applying HFL.
	16		ANN	Edge 2	yes	Unlike Edge 1 at this non-IID level, the ANN multiclass classification model achieves a slight improvement in classification performance following the application of HFL.
	17	0.7	ANN	Edge 1	No	Although the data at this non-IID level are more balanced, the ANN multiclass classification model shows a slight decrease in classification performance after applying HFL.
	18		ANN	Edge 2	No	Similarly, for Edge 2, the ANN multiclass classification model shows a slight decrease in classification performance after applying HFL.

Continued on next page

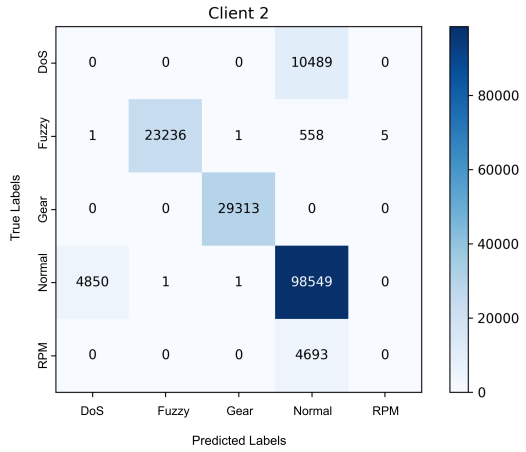
Dataset	Scenario	Non-IID level	Model	Edge	Improved	Analysis
Car Hacking: Attack & Defence Challenge 2020 Dataset [2]	19	0.3	ANN-LSTM IDS	Edge 1	No	In this highly imbalanced scenario, the ANN improves, but the overall IDS shows a slight decline in distinguishing normal from attack data after applying HFL.
	20		ANN-LSTM IDS	Edge 2	No	Similarly, in this scenario, the data are highly imbalanced, and although the ANN model shows improvement, the overall IDS exhibits a slight decrease in distinguishing between normal and attack data after applying HFL.
	21	0.5	ANN-LSTM IDS	Edge 1	yes	In this scenario, the ANN shows a slight decrease; however, the overall IDS demonstrates an improvement in distinguishing between normal and attack data.
	22		ANN-LSTM IDS	Edge 2	yes	In this scenario, although the ANN shows a slight performance decline, the overall IDS achieves a notable improvement in distinguishing normal from attack data.
	23	0.7	ANN-LSTM IDS	Edge 1	yes	In this scenario, the data are more balanced, and the overall IDS benefits from HFL, resulting in improved classification performance.
	24		ANN-LSTM IDS	Edge 2	No	In this scenario, the overall IDS shows a substantial decrease in performance after applying HFL.

The superior performance of H-FL compared with Edge-FL in our intrusion detection scenario stems from its ability to integrate complementary knowledge learned across heterogeneous, non-IID data distributions. In our setup, each edge server aggregates models from clients with varying proportions of attack and normal traffic over 50 local training rounds. This extended local training causes each edge to specialise towards its dominant local distribution; for example, an edge with attack-heavy clients becomes increasingly biased towards predicting attacks, while an edge with normal-heavy clients develops stronger capabilities for recognising legitimate traffic. As a result, each edge learns a skewed decision boundary that reflects its local data characteristics rather than the global data distribution. The critical limitation of Edge-FL is that these specialised models remain isolated. Without central aggregation, one edge's model may excel at detecting attacks but struggle with normal traffic classification, while another edge's model exhibits the opposite pattern. When deployed on diverse or balanced test data, each edge's model suffers from poor generalisation due to its bias towards local distribution characteristics. This leads to fragmented decision boundaries and inconsistent performance; Edge-FL models perform well only when test data resembles their training distribution, but degrade significantly otherwise. H-FL fundamentally addresses this limitation through central aggregation, which combines edge models into a global model that smooths and rebalances the decision boundaries learned at the edge level. By integrating attack detection expertise from attack-heavy edges with normal traffic recognition from normal-heavy edges, the global model captures a more representative approximation of the overall data distribution. This hierarchical knowledge sharing prevents overfitting to local non-IID distributions and produces a robust generalist model rather than a collection of isolated specialists, thereby yielding superior and more stable performance across diverse intrusion detection scenarios. Figures 6.13 and 6.14 present detailed confusion matrices for worked examples under different non-IID levels for the ANN multiclass model, which forms the first stage of the IDS. These matrices illustrate the number of samples for each class and the distribution of TPs, TNs, FPs, and FNs before and after applying HFL. After applying HFL, the model achieves a clearer separation between normal traffic and the various attack classes compared with the pre-HFL configuration. Similarly, Figures 6.15 and 6.16 show confusion matrices under different non-IID levels for the LSTM binary classification model, which constitutes the second stage of the IDS. Across all cases, the model benefits from HFL and exhibits improved classification performance after its application.

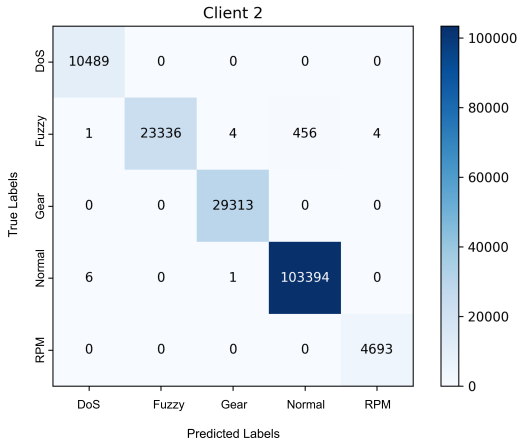


(a) Client-level confusion matrix before applying HFL.

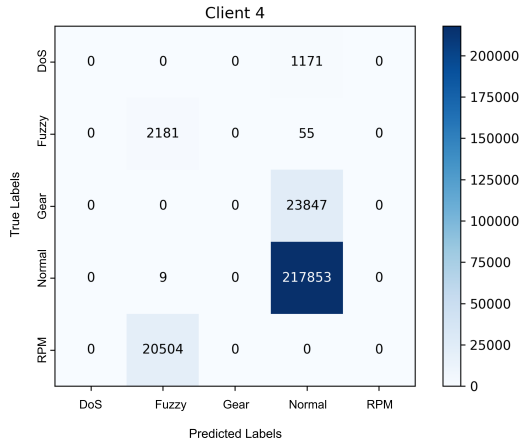
(b) Client-level confusion matrix after applying HFL.



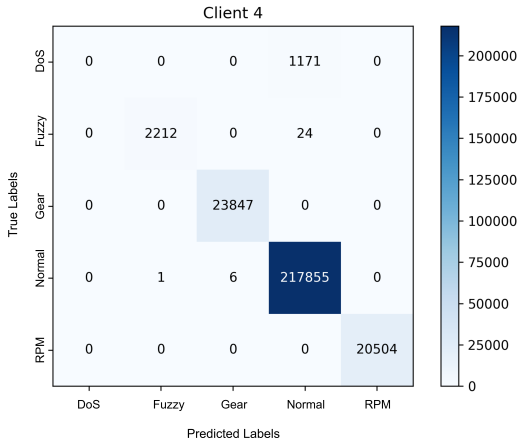
(c) Client-level confusion matrix before applying HFL.



(d) Client-level confusion matrix after applying HFL.

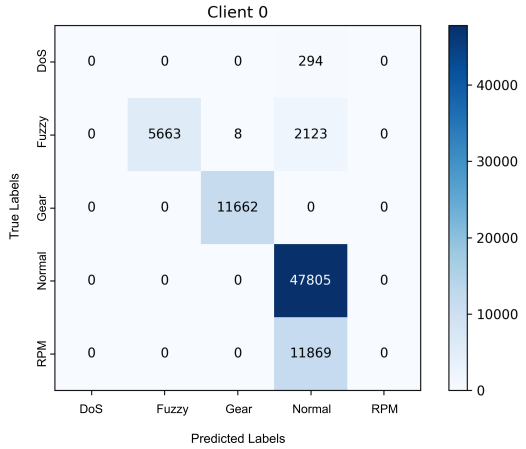


(e) Client-level confusion matrix before applying HFL.

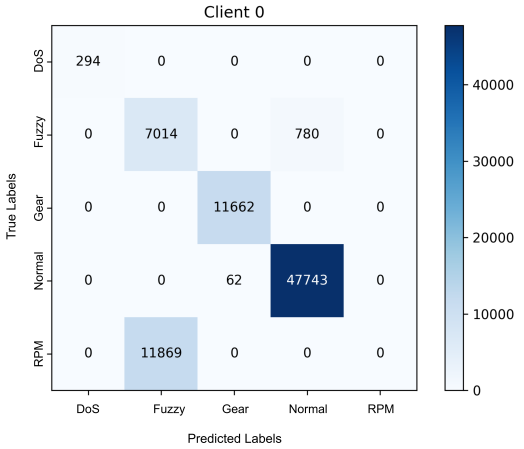


(f) Client-level confusion matrix after applying HFL.

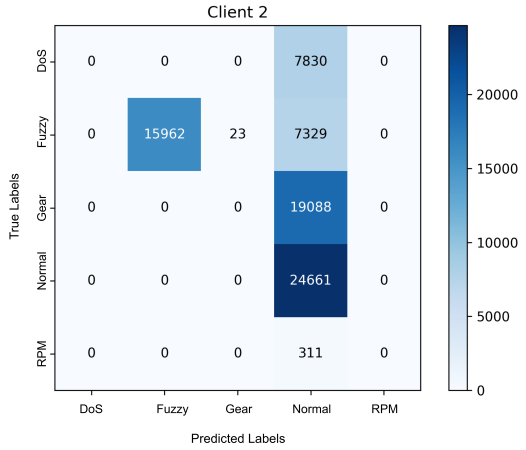
Fig. 6.13 ANN confusion matrices before and after HFL aggregation at the 0.7 non-IID level.



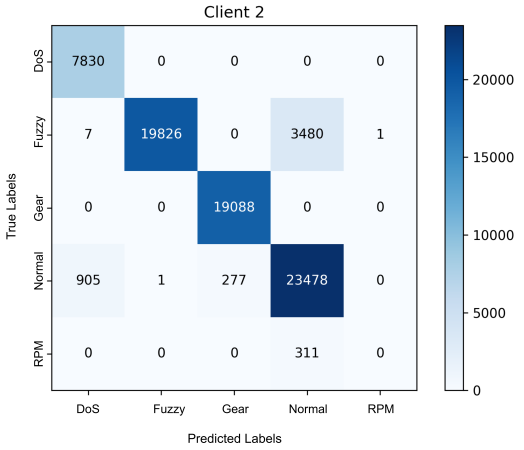
(a) Client-level confusion matrix before applying HFL.



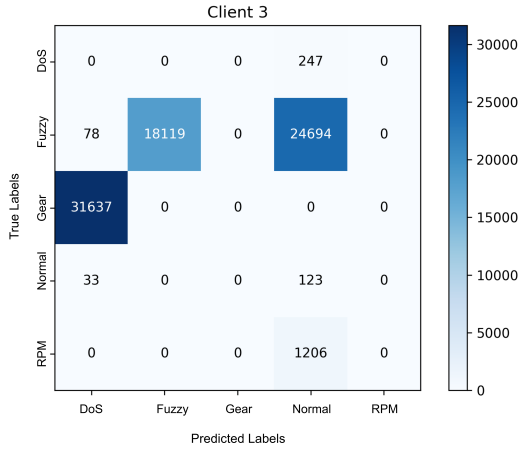
(b) Client-level confusion matrix after applying HFL.



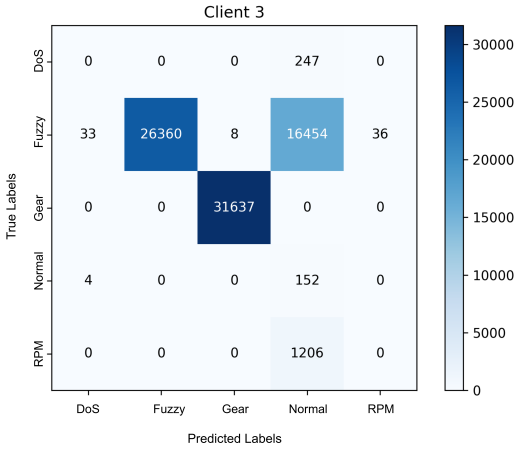
(c) Client-level confusion matrix before applying HFL.



(d) Client-level confusion matrix after applying HFL.

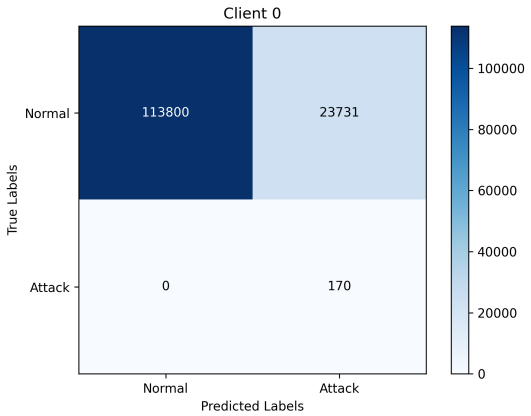


(e) Client-level confusion matrix before applying HFL.

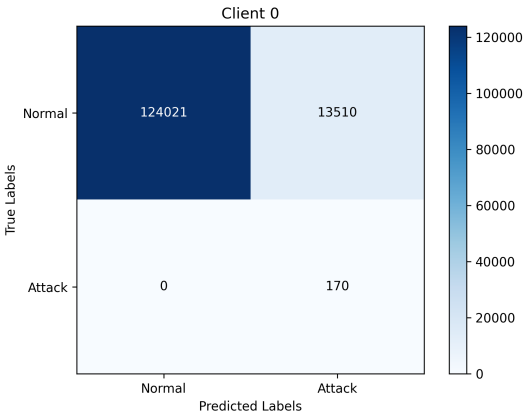


(f) Client-level confusion matrix after applying HFL.

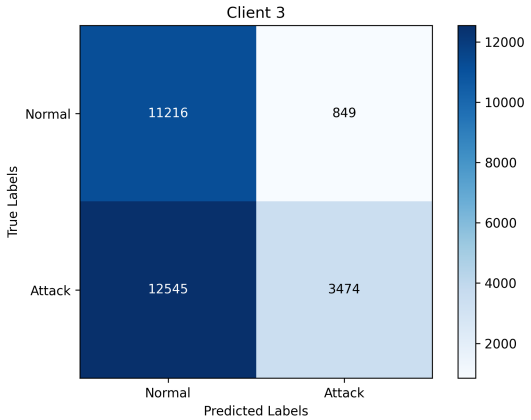
Fig. 6.14 ANN confusion matrices before and after HFL aggregation at the 0.3 non-IID level.



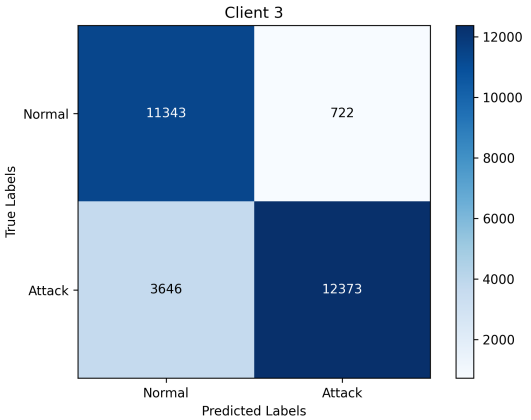
(a) Client-level confusion matrix before applying HFL.



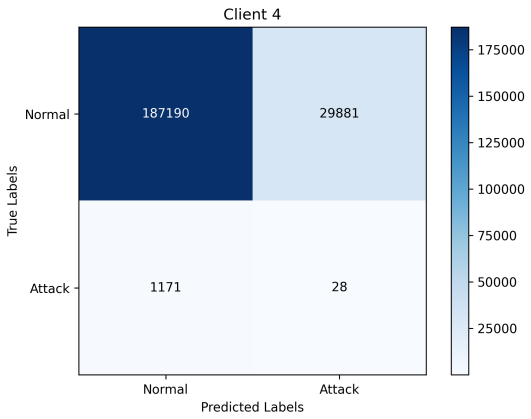
(b) Client-level confusion matrix after applying HFL.



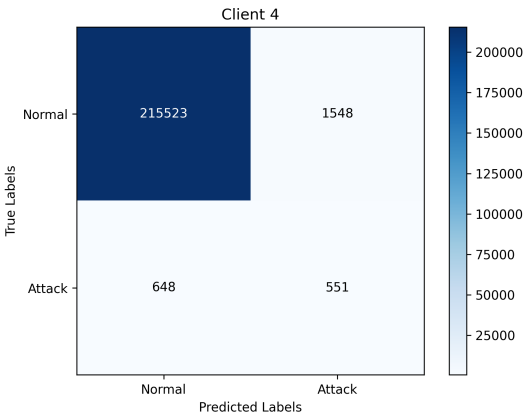
(c) Client-level confusion matrix before applying HFL.



(d) Client-level confusion matrix after applying HFL.

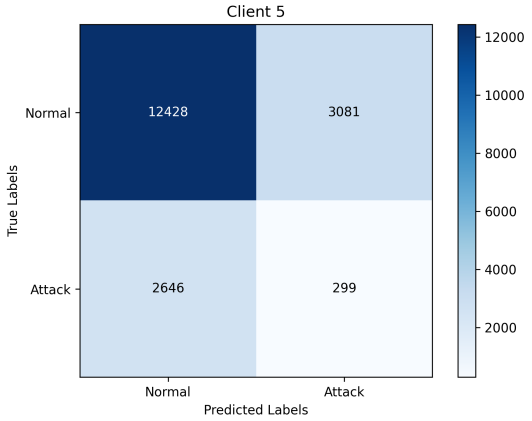


(e) Client-level confusion matrix before applying HFL.

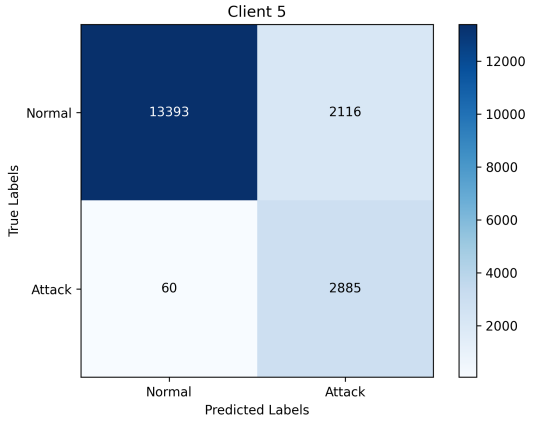


(f) Client-level confusion matrix after applying HFL.

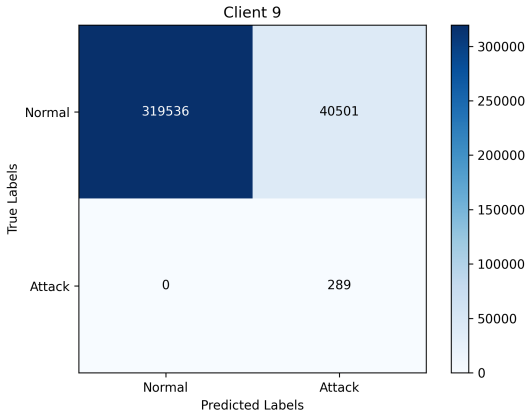
Fig. 6.15 LSTM confusion matrices before and after HFL aggregation at the 0.7 non-IID level.



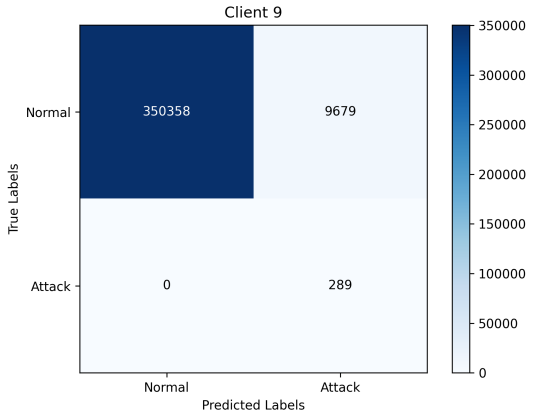
(a) Client-level confusion matrix before applying HFL.



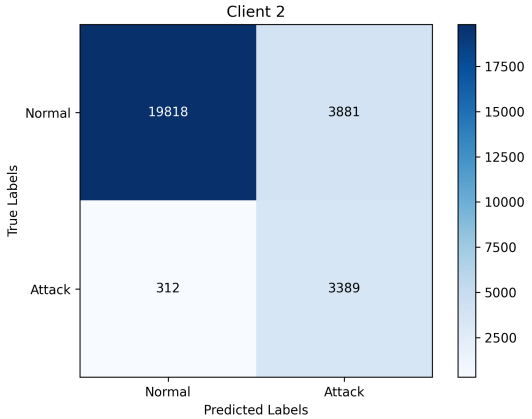
(b) Client-level confusion matrix after applying HFL.



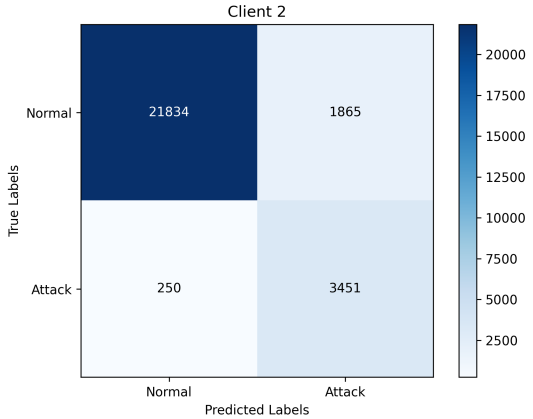
(c) Client-level confusion matrix before applying HFL.



(d) Client-level confusion matrix after applying HFL.



(e) Client-level confusion matrix before applying HFL.



(f) Client-level confusion matrix after applying HFL.

Fig. 6.16 LSTM confusion matrices before and after HFL aggregation at the 0.3 non-IID level.

Table 6.6 shows the average training time in seconds (S) for both the ANN and LSTM models, as well as the total training time for one round. The results indicate that training time depends on the number of training data, with Client 6 having the highest training time of 75.35 s due to its large training dataset, and the inverse being true for clients with smaller datasets.

Table 6.6 Average training time per client for one round.

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

We also measured the network load by calculating the size of weights transferred between the central server and edges, as well as between vehicles and edges. The size of the model weights is 0.966 MB. The network load for a single client in one round is twice the weights size, as each client sends its weights to the edge and receives the newly aggregated weights. Table 6.7 presents the network load, measured as transmitted model weights, for different FL architectures under varying numbers of clients and training rounds, highlighting communication differences between architectural designs. **Edge-FL/FL** denotes an edge server operating as a standalone FL server without aggregation with a central server, which is equivalent to a standard FL setting with one central server. **H-FL** refers to the full HFL architecture comprising the central server, edge servers (aggregators), and clients, while **Edge-H-FL** represents the edge server when it communicates with the central server as part of the hierarchical process. **FL** denotes the standard FL architecture with a single server and multiple clients. For Edge-FL/FL, a single client and one training round incur a communication cost of 1.932 MB. Introducing the central server layer transforms this configuration into H-FL and increases the transmitted model weights due to the additional communication

Table 6.7 Network load across different architectures.

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

with the central server. When the number of clients increases to five and the training rounds to 50, the Edge-FL/FL configuration results in a total transmission of 483.45 MB, which nearly doubles to 968.8 MB under the Edge-H-FL setup because of the added server level communication. In comparison, a standard FL architecture with one central server and ten clients over 50 rounds results in a communication load of 966.9 MB. Adding edge servers to this setup to form the H-FL architecture leads to a slight increase in load, totalling 970.76 MB. This illustrates that H-FL architectures tend to have a slightly higher communication cost than standard FL because the central server sends global model weights to the edges and receives aggregated weights from each edge for global aggregation. Although H-FL introduces an additional layer of communication between the central server and edges, which is absent in standard FL, it reduces bottlenecks by distributing clients across edges, dividing the network load and improving efficiency.

While the proposed H-FL mitigates several limitations of standard FL, including network congestion, communication delays, and the single point of failure associated with reliance on a single central server, these improvements arise from decentralising model aggregation across multiple edge servers. However, this architectural shift also introduces additional challenges compared to standard FL. In particular, the hierarchical design increases system complexity, as it requires the coordination and management of multiple intermediate edge servers rather than a single central server. Unlike standard FL, which relies on direct communication between clients and a central server, H-FL requires synchronisation across multiple aggregation layers, where delays or failures at intermediate edge servers can propagate through the hierarchy and affect overall training efficiency. Moreover, challenges inherent to standard FL, such as data and system heterogeneity across vehicles, persist in H-FL and are not inherently resolved by the hierarchical architecture. Finally, H-FL incurs higher training time and network load than standard FL due to the additional aggregation stages, in which the central server aggregates updates from edge servers before redistributing the global model.

Although this hierarchical aggregation enables learning from a broader range of data and attack scenarios across multiple regions, it does so at the cost of increased system overhead.

To fully assess the practicality of implementing any FL approach, it is therefore also necessary to consider the memory consumption of learning models trained locally on individual vehicles. High-end automobiles, such as those produced by Tesla, utilise advanced DL algorithms for tasks like object detection and various driver assistance features. To handle the substantial computational demands of data from cameras and sensors, these vehicles are equipped with dedicated GPUs. For example, Tesla's latest Model S and Model X are powered by AMD RDNA 2 GPUs, which provide 16–32 GB of memory for processing tasks [213]. Given that our IDS also relies on GPUs, we evaluate a vehicle's memory requirements by analysing both the size of the learning models and the memory needed to process the dataset. The total model size is 2.98 MB. To optimise memory usage during training, we process the data in batches of 256 rather than loading the entire dataset at once. This approach significantly reduces the memory required for training data from 104.25 MB (for the full dataset) to just 0.02 MB per batch. Consequently, the model requires approximately 1 GB of memory for operation, which is well within the capabilities of current vehicle-level machines.

6.5 Conclusions

The aim of this chapter was to deploy and evaluate our proposed in-vehicle IDS within the simulated H-FL framework to address the limitations of the standard FL approach. We evaluated the performance of the IDS using two datasets with three non-IID levels to simulate real-world scenarios and assess the impact of different data distributions. Experimental results indicate that the H-FL framework, in most scenarios, improves F1-score results by up to 10.63% for both known and unknown attacks across diverse non-IID conditions, outperforming edge-FL, which is constrained by smaller, vehicle-specific datasets. However, in certain scenarios, the results showed a decline after applying H-FL, potentially due to the heterogeneity of data among the clients. Overall, H-FL improved the F1-score in 16 out of 24 evaluated scenarios in two datasets. We believe that the proposed H-FL framework offers a valuable contribution to the development of FL-based IDSs in the context of in-vehicle network security. Future research should investigate how to cluster vehicles to enhance the performance of H-FL. Grouping vehicles based on factors such as geographic proximity, network connectivity, or data similarity may improve training efficiency and model performance across clients [214].

Chapter 7

Conclusions and Future Work

This chapter summarises the work conducted in this thesis, including its contributions and an overview of previous chapters. Then it redefines the research questions, linking them to the chapters that address and attempt to answer them. Finally, it highlights the research limitations and suggests potential directions for future research.

7.1 Thesis Summary and Contributions

Connected and Autonomous Vehicles (CAVs) are expected to become the backbone of future transportation systems [4], with nearly 95% of new vehicles projected to be connected to external networks by 2030 [19]. Among the various in-vehicle communication protocols, the CAN bus has emerged as the de facto standard, owing to its high speed, reliability, and ease of use [8, 7]. However, as CAVs integrate advanced connectivity and increasingly rely on software, the automotive cybersecurity threat landscape has expanded considerably [24], making these vehicles attractive targets for emerging cyberattacks [6]. The CAN protocol, for instance, was designed without built-in security features, resulting in numerous vulnerabilities that expose vehicles to significant risks. These vulnerabilities, including the lack of authentication, encryption, and network segmentation, are discussed in detail in Chapter 2. Researchers have successfully exploited these vulnerabilities in CAN bus systems to carry out cyberattacks [16, 25, 17, 18], which can compromise ECUs and lead to severe consequences. Such attacks may result in the loss of control over critical systems such as braking, steering, and acceleration, causing unexpected vehicle behaviour and, in extreme cases, loss of life [14]. Despite extensive efforts to secure communication protocols, the unique constraints and complexities of embedded systems, combined with the limitations of the CAN protocol, make many proposed defences impractical or unfeasible [26]. Given

the severity of these threats, securing the CAN bus has become a major research priority. One promising solution is the development of in-vehicle IDSs, which have demonstrated considerable effectiveness [27]. However, deploying these security measures was challenging due to the limited memory and computational resources available in vehicles, which could interfere with safety-critical operations [10, 11]. Therefore, any viable security solution must be lightweight and have a low memory footprint to ensure suitability for automotive systems. As a result, there is an urgent need for a robust yet lightweight IDS for in-vehicle networks. To address this, this thesis begins by providing insight into CAN bus data through an analysis that assesses the feasibility and effectiveness of feature selection techniques in building a robust IDS for in-vehicle networks. Based on this analysis, it proposes a robust, lightweight (low memory footprint), multi-stage IDS capable of detecting both known and previously unseen attacks. To preserve data privacy, leverage information from a broader range of vehicles, adapt to new unknown attacks, and mitigate the risk of single-point failure, the proposed IDS is deployed and evaluated within a simulated H-FL environment.

In Chapter 1, the thesis begins with a brief overview of CAVs, highlighting their significance and prominence in the future of transportation, as well as the security challenges they face. The research problem is defined in detail, along with the motivation behind this work. Additionally, the chapter outlines the key contributions made throughout this thesis and the publications derived from it.

In Chapter 2, we provided an overview of in-vehicle protocols, focusing on the CAN bus protocol and emphasising its critical role in in-vehicle communication. We also presented the main CAN bus vulnerabilities, including the lack of authentication, encryption, network segmentation, broadcast domain issues, ID-based priority, and external interfaces. Following this, we explored various entry points to the in-vehicle network and presented various attack scenarios, along with the consequences of such attacks. This chapter serves as a foundational background, highlighting the severity of the problem and its potential consequences. It underscores the urgent need to enhance the security of in-vehicle networks.

In Chapter 3, we provide a comprehensive review of current ML and DL-based in-vehicle IDSs designed to detect known, unknown, and combined known–unknown attacks, as well as FL-based in-vehicle IDSs. To ensure comprehensive coverage of the literature, we followed Kitchenham's [82] search method to gather relevant studies. This chapter aimed to identify limitations and research gaps in existing work. Based on the reviewed literature and identified limitations, to the best of our knowledge, no prior work has specifically explored feature selection for CAN bus data. Additionally, no previous research has proposed a comprehensive in-vehicle IDS capable of detecting both known and unknown attacks using DL algorithms,

while maintaining a lightweight model size and deploying the IDS within the simulated H-FL framework. Seven research questions were formulated based on these identified gaps, and the subsequent chapters address each of them.

In Chapter 4, we provide insight into CAN bus data by analysing it to assess the feasibility and effectiveness of feature selection techniques in building a robust IDS for in-vehicle networks, demonstrating that each CAN ID exhibits unique data patterns, meaning that features important for one CAN ID may not be relevant for another. As a result, it is impractical to select a consistent set of important features across all CAN IDs. Additionally, we compared the performance of various feature selection methods, revealing that nearly all comparable methods identified some or all zero-constant features as important, despite these features being intended to handle irrelevant information. This suggests that applying feature selection techniques uniformly to all CAN IDs may result in the loss of valuable information.

In Chapter 5, we designed a robust in-vehicle DL-based IDS that addresses the limitations of previous works. When designing an in-vehicle IDS, several critical decisions were made during the design phase that were overlooked in many proposed IDSs. One key decision was to include both CAN ID and payload data (CAN frame), enabling the detection of CAN ID changes and payload manipulation attacks [27]. Moreover, unlike traditional ML approaches, we did not apply feature selection techniques for several reasons: DL automatically extracts features, allowing algorithms to discern optimal features directly from raw data [167]. Additionally, attackers may exploit neglected features in the future [164, 165]. Most importantly, our findings from the first research question indicate that feature selection is not efficient for CAN bus data. The proposed IDS employs a multi-stage approach: an ANN in the first stage to detect seen attacks, and an LSTM autoencoder in the second stage to detect new, 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 first model, providing an additional layer of protection. We trained and tested our IDS using a benchmark dataset based on real-world traffic data [1], which is extensively used in automotive security research. Experimental results showed that our IDS achieves an F1-score exceeding 0.99 for seen attacks and exceeding 0.95 for unseen attacks, with a detection rate of 99.99%. Additionally, the False Alarm Rate (FAR) is exceptionally low at 0.016%, minimising false alarms. This makes our model robust against seen and unseen attacks. Moreover, the development and deployment of IDSs are significantly impacted by the constraints of Electronic Control Units (ECUs) in-vehicle networks, which include limited memory storage, computing power, and bandwidth [27]. Despite employing DL

algorithms, our IDS remains lightweight, ensuring its feasibility for real-world deployment. In pursuit of optimal results, we have simplified the model architecture to minimise its size. The sizes of the ANN and LSTM autoencoder models were calculated to be 0.030 MB and 2.95 MB, respectively, resulting in a combined total of 2.98 MB. Moreover, models with fewer parameters offer the advantage of faster training and testing times [63]. In our proposed IDS, the ANN model has 517 trainable parameters, while the LSTM autoencoder includes 253,065, resulting in a combined total of 253,582 trainable parameters. This is significantly fewer than those used in existing approaches, further enhancing the model's efficiency. Moreover, since CAN is a time-critical system, inference time and detection latency are essential safety-related metrics for in-vehicle IDS to ensure real-time performance. Our proposed IDS achieves a latency of less than 0.04 ms, meeting the real-time requirements of vehicle safety services. This chapter demonstrates the feasibility of developing an in-vehicle IDS capable of detecting and classifying both known and unknown attacks by leveraging the advantages of DL algorithms, while maintaining a compact model size, a low number of trainable parameters, and low latency.

In Chapter 6, we deployed and evaluated the IDS proposed in Chapter 5 within the simulated H-FL framework to generalise across diverse driving behaviours and detect new attacks beyond the server's coverage, while mitigating the risk of single-point failure inherent in standard FL architectures. We evaluated the performance of the IDS using two datasets with three levels of non-IID data distributions to simulate real-world scenarios and assess the impact of different data distributions. Experimental results indicate that the H-FL framework, in most scenarios, improves F1-score results by up to 10.63% for both known and unknown attacks across diverse non-IID conditions, outperforming edge-FL, which is constrained by smaller, vehicle-specific datasets. However, in certain scenarios, the results showed a decline after applying H-FL, potentially due to the heterogeneity of data among the clients. Overall, H-FL improved the F1-score in 16 out of 24 evaluated scenarios in two CAN bus datasets.

To conclude, this thesis examined the growing threats of cyberattacks on the CAN bus, the primary in-vehicle communication protocol, and developed a novel, robust IDS that addresses the limitations of previous approaches. The proposed IDS not only detects and classifies known attacks but is also prepared for future threats by employing an anomaly detection model to identify new, unknown attacks or those that evade the first model. Unlike other existing works, our proposed IDS consists of two cascaded models, where the second model re-evaluates the traffic data classified as normal in the first model to detect any malicious traffic that evades the first detection model. To further enhance the system and leverage diverse driving behaviours while preserving data privacy, we utilised an FL environment

instead of centralised training. Additionally, to overcome the limitations of traditional FL architecture, we proposed and evaluated our IDS within an H-FL environment.

7.2 Research Questions Answered

In this section, we restate the research questions outlined in Chapter 3. We then present responses to each question and describe the research efforts undertaken in this thesis to address them as follows:

- **Research Question 1:** *Do all CAN IDs share the same important payload features, or should all CAN bus features be considered to ensure system reliability?*

  In Chapter 4, we analysed CAN bus data from three different datasets and observed that, for each CAN ID, payload values can be zero-constant, constant non-zero, or variable, depending on how the CAN ID is defined in the dictionary. As a result, features that are important for one CAN ID may be zero-constant for others.

- **Research Question 2:** *Do feature selection techniques identify important features in CAN bus data for IDS design?*

  In Chapter 4, we apply six different feature selection techniques—including univariate selection, extra trees classifier, information gain, RFE, chi-square test, and Pearson correlation—to a sample of CAN IDs from each dataset to assess how these techniques handle the zero-constant features. This analysis and evaluation of different feature selection methods help assess the effectiveness of these techniques in the context of the CAN bus.

- **Research Question 3:** *How can an in-vehicle IDS be designed to detect both known and unknown attacks using DL algorithms?*

  In Chapter 5, we developed a multi-stage IDS capable of detecting both known and unknown attacks using DL algorithms. To ensure that the IDS is lightweight and meets the resource constraints in vehicles, we kept the models as simple as possible while maintaining a high detection rate. We measured the proposed IDS's size in MB and the number of trainable parameters, comparing it with other works that use traditional ML algorithms to ensure that it meets the in-vehicle IDS requirements.

- **Research Question 4:** *How can the IDS remain lightweight and meet real-time requirements while operating within resource constraints?*

In Chapter 5, we developed an in-vehicle IDS using CAN frame data (CAN ID and payload) without employing feature selection techniques, based on our findings from RQ1 and RQ2. To enhance the robustness of the proposed IDS, it can detect and classify known attacks by type, as well as identify unknown attacks. Moreover, the proposed IDS is updated as new unknown attacks are detected. It is designed as a multi-stage system, where the second model re-examines the data classified as normal in the first model to detect unseen attacks that bypassed the initial model, providing an additional layer of protection. To evaluate the performance of the proposed IDS, we used two benchmark datasets.

- **Research Question 5:** *How can in-vehicle IDSs detect new attacks that were never part of the central server's coverage in the FL architecture?*

  In Chapter 6, we proposed an H-FL framework that incorporates multiple edge aggregators alongside the central aggregator, rather than relying on one central aggregator for all clients (vehicles). Each edge will cover a group of vehicles under its coverage. The models from these edge aggregators are then transmitted to the central server for global aggregation, thereby enhancing the system's ability to generalise and detect previously unseen attacks. We evaluated the performance of our proposed IDS in this H-FL environment using two CAN bus datasets, considering realistic non-IID data when distributing the data across clients.

- **Research Question 6:** *How can the architecture of FL-based in-vehicle IDSs be enhanced to mitigate network congestion and communication delays?*

  In Chapter 6, we proposed an H-FL framework that integrates multiple edge aggregators alongside the central aggregator, rather than relying solely on a single central server. By distributing clients across different edges, the framework not only balances the communication load but also reduces the likelihood of bottlenecks that commonly arise when all vehicles communicate directly with one central server. As a result, the system achieves more efficient use of network resources, lowers delays in transmitting updates, and improves the overall scalability of the IDS.

- **Research Question 7:** *How can we reduce the risk of single-point failure inherent in centralised FL architectures?*

  In Chapter 6, we proposed an H-FL framework that integrates multiple edge aggregators alongside the central aggregator, rather than relying solely on a single central server. By distributing aggregation across several edge nodes, each responsible for a subset

of vehicles, the framework mitigates the dependency on one central component. This design not only broadens the diversity of training data but also reduces vulnerability to single-point failures, thereby enhancing the robustness and resilience of the overall system.

7.3 Limitations and Future Research Directions

This thesis provides a strong foundation for future research aimed at improving the detection capabilities and robustness of the proposed in-vehicle IDS. However, several limitations identified in this work open up multiple avenues for further investigation and enhancement.

- **Limited Access to Real-World Datasets:** A key limitation of this research is the restricted access to real-world datasets that capture diverse driving behaviours and environments, such as urban, mountainous, and rural terrains. Real-world driving conditions are far more complex than those represented in the available datasets. The primary reason for the lack of such datasets is privacy and legal issues [125]. As a result, the proposed IDS has been trained and evaluated under constrained conditions, limiting its ability to model normal vehicle behaviour comprehensively across varied scenarios. One promising future direction is exploring streaming learning, which allows the model to dynamically adjust in real-time as the vehicle encounters different driving environments. This approach could improve the system's ability to adapt and enhance detection accuracy in a broader range of conditions.

- **Vehicle-Specific Models and Generalisation Challenges:** Another limitation is the assumption that all vehicles in the FL environment share the same make, model, CAN IDs, and payload interpretations. This assumption could necessitate developing separate models for each vehicle make and model, leading to increased complexity. Generalising the IDS to learn across different vehicle types, rather than relying on distinct models for each, remains a significant challenge due to variations in CAN bus data and the lack of access to DBC files, which define signal meanings. While FL has shown promise in enhancing IDS performance by integrating models from diverse driving scenarios and vehicle states, achieving robust model generalisation across all vehicle types is complex. Future research could explore techniques such as domain adaptation or transfer learning to bridge the gap between different vehicle models and make the system more general across all vehicle types.

- **Limited known Attack Types:** Another limitation is the limited number of available known attacks, despite the model being trained on four attack types present in the dataset. As new attacks continue to emerge, identifying and incorporating additional attack signatures into the signature-based model is essential to narrowing the attack surface. In the absence of a global attack signature database, a viable solution is to establish a dedicated repository for CAN bus malware to enable more effective protection and countermeasures. However, since ECUs vary across manufacturers, each Original Equipment Manufacturer (OEM), such as Mercedes, Toyota, or BMW, would need to build, update, and maintain a separate database for their own vehicles, potentially hosted in an OEM-specific cloud [12].

- **False Positives in Unsupervised Learning:** As with all unsupervised learning methods [27], the anomaly detector model in this research produces false positives. In critical systems, minimising false alarms is essential for maintaining system reliability. Some existing approaches train biased classifiers to reduce false positives and false negatives, but this shifts the model away from being purely unsupervised. Future research should focus on finding practical solutions that reduce false positives without compromising the model's unsupervised nature. One potential direction is to leverage eXplainable AI (XAI) techniques to make the behaviour of in-vehicle IDSs more interpretable and transparent. While AI methods have shown great potential in combating cyberattacks, they often generate false alarms and produce decisions that are difficult to interpret, leading to uncertainty and distrust [215]. XAI methods, such as SHapley Additive exPlanations (SHAP) or Local Interpretable Model-agnostic Explanations (LIME), can provide clearer insights into the decision-making process of IDSs, allowing for better responses to alarms and fostering greater trust in AI-driven security systems [216]. Further exploration of XAI could significantly improve both the transparency and reliability of AI-based in-vehicle IDSs. Another important direction is to train ML-based IDSs using data collected across diverse driving conditions, including while parked, idling, and in motion [167].

- **Client Selection in FL:** In this thesis, we assumed that all vehicles are reachable and available to participate in the FL training. A potential direction for future work is to explore methods for client selection or exclusion to improve the efficiency of the FL process. Investigating effective client selection strategies is crucial to optimising model accuracy while minimising computational and communication overhead. Potential

strategies could involve selecting clients based on similarities in CAN bus data, driving behaviour, or geographical area to ensure that the FL process remains efficient.

References

[1] Eunbi Seo, Hyun Min Song, and Huy Kang Kim. Gids: Gan based intrusion detection system for in-vehicle network. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6. IEEE, 2018.

[2] Hyunjae Kang, Byung Il Kwak, Young Hun Lee, Haneol Lee, Hwejae Lee, and Huy Kang Kim. Car hacking and defense competition on in-vehicle network. In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, volume 2021, page 25, 2021.

[3] Numaan Huq and Rainer Vosseler. Identifying cybersecurity focus areas in connected cars based on wp. 29 un-r155 attack vectors and beyond. Technical report, SAE Technical Paper, 2022.

[4] Fatimah Aloraini, Amir Javed, and Omer Rana. Adversarial attacks on intrusion detection systems in in-vehicle networks of connected and autonomous vehicles. *Sensors*, 24(12):3848, 2024.

[5] SMMT Driving the Motor Industry. Connected and autonomous vehicles: The global race to market. Technical report, THE SOCIETY OF MOTOR MANUFACTURERS AND TRADERS LIMITED, 2019.

[6] James Pickford, Rasadhi Attale, Siraj Shaikh, Hoang Nga Nguyen, and Lee Harrison. Systematic risk characterisation of hardware threats to automotive systems. *Journal on Autonomous Transportation Systems*, 1(4):1–36, 2024.

[7] Omar Y Al-Jarrah, Carsten Maple, Mehrdad Dianati, David Oxtoby, and Alex rMouzakitis. Intrusion detection systems for intra-vehicle networks: A review. *Ieee Access*, 7:21266–21289, 2019.

[8] Muzun Althunayyan, Amir Javed, and Omer Rana. A robust multi-stage intrusion detection system for in-vehicle network security using hierarchical federated learning. *Vehicular Communications*, 49:100837, 2024.

[9] Avishek Paul and Md Rabiul 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)*, pages 1–5. IEEE, 2021.

[10] Mert D Pesé, Jay W Schauer, Junhui Li, and Kang G Shin. S2-can: Sufficiently secure controller area network. In *Proceedings of the 37th Annual Computer Security Applications Conference*, pages 425–438, 2021.

[11] Ali Barati, Ali Movaghar, and Masoud Sabaei. Energy efficient and high speed error control scheme for real time wireless sensor networks. *International Journal of Distributed Sensor Networks*, 10(5):698125, 2014.

[12] Emad Aliwa, Omer Rana, Charith Perera, and Peter Burnap. Cyberattacks and countermeasures for in-vehicle networks. *ACM computing surveys (CSUR)*, 54(1):1–37, 2021.

[13] Upstream Security Ltd. Upstream's 2024 global automotive cybersecurity report. https://upstream.auto/reports/global-automotive-cybersecurity-report/#, 2024. Accessed: 2025-01-18.

[14] Clinton Young, Joseph Zambreno, Habeeb Olufowobi, and Gedare Bloom. Survey of automotive controller area network intrusion detection systems. *IEEE Design & Test*, 36(6):48–55, 2019.

[15] Ken Tindell. Can injection: keyless car theft. https://kentindell.github.io/2023/04/03/can-injection/, 2023. Accessed: 2025-01-18.

[16] Jordan 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, 2016. (accessed 1 April 2023).

[17] Tencent Keen Security 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/, 2018. (accessed 10 April 2023).

[18] Tencent Keen Security 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/, 2020. (accessed 10 April 2023).

[19] Michele Bertoncello, Christopher Martens, Timo Möller, and Tobias 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, 2021. (accessed 6 April 2023).

[20] Aditya Vikram et al. Anomaly detection in network traffic using unsupervised machine learning approach. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pages 476–479. IEEE, 2020.

[21] Siti-Farhana Lokman, Abu Talib Othman, and Muhammad-Husaini Abu-Bakar. Intrusion detection system for automotive controller area network (can) bus system: a review. *EURASIP Journal on Wireless Communications and Networking*, 2019:1–17, 2019.

[22] Khalid Mohammed Almatar. Traffic congestion patterns in the urban road network:(dammam metropolitan area). *Ain Shams engineering journal*, 14(3):101886, 2023.

[23] Md Mokhlesur Rahman and Jean-Claude Thill. Impacts of connected and autonomous vehicles on urban transportation and environment: A comprehensive review. *Sustainable Cities and Society*, 96:104649, 2023.

[24] Riccardo Coppola and Maurizio Morisio. Connected car: technologies, issues, future trends. *ACM Computing Surveys (CSUR)*, 49(3):1–36, 2016.

[25] Jeff Crume. Ownstar: Yet another car hack. https://insideinternetsecurity.wordpress.com/2015/08/05/ownstar-yet-another-car-hack/, 2015. (accessed 20 March 2023).

[26] Valliappa Chockalingam, Ian Larson, Daniel Lin, and Spencer Nofzinger. Detecting attacks on the can protocol with machine learning. *Annu EECS*, 558(7), 2016.

[27] Sampath Rajapaksha, Harsha Kalutarage, M Omar Al-Kadri, Andrei Petrovski, Garikayi Madzudzo, and Madeline Cheah. Ai-based intrusion detection systems for in-vehicle networks: A survey. *ACM Computing Surveys*, 55(11):1–40, 2023.

[28] Qingling Zhao, Mingqiang Chen, Zonghua Gu, Siyu Luan, Haibo Zeng, and Samarjit Chakrabory. Can bus intrusion detection based on auxiliary classifier gan and out-of-distribution detection. *ACM Transactions on Embedded Computing Systems (TECS)*, 21(4):1–30, 2022.

[29] Vishnu Pandi Chellapandi, Liangqi Yuan, Stanislaw H Żak, and Ziran Wang. A survey of federated learning for connected and automated vehicles. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2485–2492. IEEE, 2023.

[30] Edy Kristianto, Po-Ching Lin, and Ren-Hung Hwang. Sustainable and lightweight domain-based intrusion detection system for in-vehicle network. *Sustainable Computing: Informatics and Systems*, 41:100936, 2024.

[31] Linxi Zhang, Xuke Yan, and Di Ma. Efficient and effective in-vehicle intrusion detection system using binarized convolutional neural network. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pages 2299–2307. IEEE, 2024.

[32] Vipin Kumar Kukkala, Sooryaa Vignesh Thiruloga, and Sudeep Pasricha. Indra: Intrusion detection using recurrent autoencoders in automotive embedded systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):3698–3710, 2020.

[33] Sooryaa Vignesh Thiruloga, Vipin Kumar Kukkala, and Sudeep Pasricha. Tenet: Temporal cnn with attention for anomaly detection in automotive cyber-physical systems. In *2022 27th Asia and South Pacific design automation conference (ASP-DAC)*, pages 326–331. IEEE, 2022.

[34] Jadil Alsamiri and Khalid Alsubhi. Federated learning for intrusion detection systems in internet of vehicles: A general taxonomy, applications, and future directions. *Future Internet*, 15(12):403, 2023.

[35] Maha Driss, Iman Almomani, Zil e Huma, and Jawad Ahmad. A federated learning framework for cyberattack detection in vehicular sensor networks. *Complex & Intelligent Systems*, 8(5):4221–4235, 2022.

[36] Kabid Hassan Shibly, Md Delwar Hossain, Hiroyuki Inoue, Yuzo Taenaka, and Youki Kadobayashi. Personalized federated learning for automotive intrusion detection systems. In *2022 IEEE Future Networks World Forum (FNWF)*, pages 544–549. IEEE, 2022.

[37] Tianqi Yu, Guodong Hua, Huaisheng Wang, Jianfeng Yang, and Jianling Hu. Federated-lstm based network intrusion detection method for intelligent connected vehicles. *IEEE International Conference on Communications (ICC)*, 2022.

[38] Hengrun Zhang, Kai Zeng, and Shuai Lin. Federated graph neural network for fast anomaly detection in controller area networks. *IEEE Transactions on Information Forensics and Security*, 18:1566–1579, 2023.

[39] Jianfeng Yang, Jianling Hu, and Tianqi Yu. Federated ai-enabled in-vehicle network intrusion detection for internet of vehicles. *Electronics*, 2022.

[40] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

[41] Omer Rana, Theodoros Spyridopoulos, Nathaniel Hudson, Matt Baughman, Kyle Chard, Ian Foster, and Aftab Khan. Hierarchical and decentralised federated learning. In *2022 Cloud Continuum*, pages 1–9. IEEE, 2022.

[42] Bilal Jan, Haleem Farman, Murad Khan, Muhammad Imran, Ihtesham Ul Islam, Awais Ahmad, Shaukat Ali, and Gwanggil Jeon. Deep learning in big data analytics: a comparative study. *Computers & Electrical Engineering*, 75:275–287, 2019.

[43] Muzun Althunayyan, Amir Javed, and Omer Rana. An innovative feature selection approach for can bus data leveraging constant value analysis. In *International Conference on Cyber Security, Privacy in Communication Networks*, pages 69–84. Springer, 2023.

[44] Muzun Althunayyan, Amir Javed, Omer Rana, and Theodoros Spyridopoulos. Hierarchical federated learning-based intrusion detection for in-vehicle networks. *Future Internet*, 16(12):451, 2024.

[45] Muzun Althunayyan, Amir Javed, and Omer Rana. A survey of learning-based intrusion detection systems for in-vehicle networks. *Computer Networks*, page 112031, 2026.

[46] Trupil Limbasiya, Ko Zheng Teng, Sudipta Chattopadhyay, and Jianying Zhou. A systematic survey of attack detection and prevention in connected and autonomous vehicles. *Vehicular Communications*, 37:100515, 2022.

[47] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief. Client-edge-cloud hierarchical federated learning. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020.

[48] Usman Ahmad, Mu Han, Alireza Jolfaei, Sohail Jabbar, Muhammad Ibrar, Aiman Erbad, Houbing Herbert Song, and Yazeed Alkhrijah. A comprehensive survey and tutorial on smart vehicles: Emerging technologies, security issues, and solutions using machine learning. *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[49] Safa Boumiza and Rafik Braham. Intrusion threats and security solutions for autonomous vehicle networks. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 120–127. IEEE, 2017.

[50] B Vinodh Kumar and J Ramesh. Automotive in vehicle network protocols. In *2014 International Conference on Computer Communication and Informatics*, pages 1–5. IEEE, 2014.

[51] Hamideh Taslimasa, Sajjad Dadkhah, Euclides Carlos Pinto Neto, Pulei Xiong, Suprio Ray, and Ali A Ghorbani. Security issues in internet of vehicles (iov): A comprehensive survey. *Internet of Things*, 22:100809, 2023.

[52] Paul Carsten, Todd R Andel, Mark Yampolskiy, and Jeffrey T McDonald. In-vehicle networks: Attacks, vulnerabilities, and proposed solutions. In *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, pages 1–8, 2015.

[53] Jiajia Liu, Shubin Zhang, Wen Sun, and Yongpeng Shi. In-vehicle network attacks and countermeasures: Challenges and future directions. *IEEE Network*, 31(5):50–58, 2017.

[54] Guillaume Dupont, Jerry den Hartog, Sandro Etalle, and Alexios Lekidis. A survey of network intrusion detection systems for controller area network. In *2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 1–6. IEEE, 2019.

[55] Wufei Wu, Renfa Li, Guoqi Xie, Jiyao An, Yang Bai, Jia Zhou, and Keqin Li. A survey of intrusion detection for in-vehicle networks. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):919–933, 2019.

[56] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *20th USENIX Security Symposium (USENIX Security 11)*, San Francisco, CA, 2011. USENIX Association.

[57] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, et al. Experimental security analysis of a modern automobile. In *2010 IEEE symposium on security and privacy*, pages 447–462. IEEE, 2010.

[58] Samuel Woo, Hyo Jin Jo, and Dong Hoon Lee. A practical wireless attack on the connected car and security protocol for in-vehicle can. *IEEE Transactions on intelligent transportation systems*, 16(2):993–1006, 2014.

[59] Daniel S Fowler, Jeremy Bryans, Siraj Ahmed Shaikh, and Paul Wooderson. Fuzz testing for automotive cyber-security. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 239–246. IEEE, 2018.

[60] Kazuki Iehira, Hiroyuki Inoue, and Kenji Ishida. Spoofing attack using bus-off attacks against a specific ecu of the can bus. In *2018 15th IEEE annual consumer communications & networking conference (CCNC)*, pages 1–4. IEEE, 2018.

[61] Hyunsung Lee, Seong Hoon Jeong, and Huy Kang 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)*, pages 57–5709. IEEE, 2017.

[62] Zaina Abuabed, Ahmad Alsadeh, and Adel Taweel. Stride threat model-based framework for assessing the vulnerabilities of modern vehicles. *Computers & Security*, 133:103391, 2023.

[63] Thien-Nu Hoang and Daehee Kim. Detecting in-vehicle intrusion via semi-supervised learning-based convolutional adversarial autoencoders. *Vehicular Communications*, 38, 2022.

[64] Kyong-Tak Cho and Kang G Shin. Error handling of in-vehicle networks makes them vulnerable. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1044–1055, 2016.

[65] Md Delwar Hossain, Hiroyuki Inoue, Hideya Ochiai, Doudou Fall, and Youki Kadobayashi. An effective in-vehicle can bus intrusion detection system using cnn deep learning approach. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6. IEEE, 2020.

[66] Md Delwar Hossain, Hiroyuki Inoue, Hideya Ochiai, Doudou Fall, and Youki Kadobayashi. Lstm-based intrusion detection system for in-vehicle can bus communications. *IEEE Access*, 8:185489–185502, 2020.

[67] Charlie Miller. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015.

[68] Jürgen Dürrwang, Johannes Braun, Marcel Rumez, Reiner Kriesten, and Alexander Pretschner. Enhancement of automotive penetration testing with threat analyses results. *SAE International Journal of Transportation Cybersecurity and Privacy*, 1(11-01-02-0005):91–112, 2018.

[69] Jay Nagarajan, Pegah Mansourian, Muhammad Anwar Shahid, Arunita Jaekel, Ikjot Saini, Ning Zhang, and Marc Kneppers. Machine learning based intrusion detection systems for connected autonomous vehicles: A survey. *Peer-to-Peer Networking and Applications*, pages 1–33, 2023.

[70] Jose L Hernandez-Ramos, Georgios Karopoulos, Efstratios Chatzoglou, Vasileios Kouliaridis, Enrique Marmol, Aurora Gonzalez-Vidal, and Georgios Kambourakis. Intrusion detection based on federated learning: a systematic review. *arXiv preprint arXiv:2308.09522*, 2023.

[71] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Applying intrusion detection to automotive it-early insights and remaining challenges. *Journal of Information Assurance and Security (JIAS)*, 4(6):226–235, 2009.

[72] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[73] Shaashwat Agrawal, Sagnik Sarkar, Ons Aouedi, Gokul Yenduri, Kandaraj Piamrat, Mamoun Alazab, Sweta Bhattacharya, Praveen Kumar Reddy Maddikunta, and Thippa Reddy Gadekallu. Federated learning for intrusion detection system: Concepts, challenges and future directions. *Computer Communications*, 195:346–361, 2022.

[74] Rodolfo Stoffel Antunes, Cristiano André da Costa, Arne Küderle, Imrana Abdullahi Yari, and Björn Eskofier. Federated learning for healthcare: Systematic review and architecture proposal. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–23, 2022.

[75] Ahmed Imteaj and M Hadi Amini. Leveraging asynchronous federated learning to predict customers financial distress. *Intelligent Systems with Applications*, 14:200064, 2022.

[76] Mamoun Alazab, Swarna Priya Rm, Praveen Kumar Reddy Maddikunta, Thippa Reddy Gadekallu, Quoc-Viet Pham, et al. Federated learning for cyber-

security: Concepts, challenges, and future directions. *IEEE Transactions on Industrial Informatics*, 18(5):3501–3509, 2021.

[77] Official Journal of the European Union. Regulation of the european parliament and of the council on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive (general data protection regulation). https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679, 2016. (accessed 2 April 2025).

[78] State of California Department of Justice. California consumer privacy act (ccpa). https://oag.ca.gov/privacy/ccpa, 2024. (accessed 8 Febreuary 2025).

[79] Office of the Privacy Commissioner of Canada. The personal information protection and electronic documents act (pipeda). https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/, 2021. (accessed 15 April 2025).

[80] Data Protection Management Solutions. General personal data protection act (lgpd). https://lgpd-brazil.info/, 2018. (accessed 15 April 2025).

[81] X.1375 Working Group. Guidelines for an intrusion detection system for in-vehicle networks. Technical Report X.1375, International Telecommunication Union, 2020.

[82] Barbara Kitchenham and Pearl Brereton. A systematic review of systematic review process research in software engineering. *Information and software technology*, 55(12):2049–2075, 2013.

[83] Nada Alhirabi, Omer Rana, and Charith Perera. Security and privacy requirements for the internet of things: A survey. *ACM Transactions on Internet of Things*, 2(1):1–37, 2021.

[84] Claes Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, pages 1–10, 2014.

[85] Hyun Min Song, Jiyoung Woo, and Huy Kang Kim. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, 21:100198, 2020.

[86] Rafi Ud Daula Refat, Abdulrahman Abu Elkhail, Azeem Hafeez, and Hafiz Malik. Detecting can bus intrusion by applying machine learning method to graph based features. In *Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys) Volume 3*, pages 730–748. Springer, 2022.

[87] Srinivasa Rao Nandam, Adouthu Vamshi, and Inapanuri Sucharitha. Can intrusion detection using long short-term memory (lstm). In *Computer Communication, Networking and IoT: Proceedings of 5th ICICC 2021, Volume 2*, pages 295–302. Springer, 2022.

[88] Auangkun Rangsikunpum, Sam Amiri, and Luciano Ost. An fpga-based intrusion detection system using binarised neural network for can bus systems. In *2024 IEEE International Conference on Industrial Technology (ICIT)*, pages 1–6. IEEE, 2024.

[89] Yuxi Wu and Xiaodong Tao. Network traffic anomaly detection in can bus based on ensemble learning. In *2024 4th International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, pages 240–245. IEEE, 2024.

[90] Min-Joo Kang and Je-Won Kang. Intrusion detection system using deep neural network for in-vehicle network security. *PloS one*, 11(6):e0155781, 2016.

[91] Fabio Martinelli, Francesco Mercaldo, Vittoria Nardone, and Antonella Santone. Car hacking identification through fuzzy logic algorithms. In *2017 IEEE international conference on fuzzy systems (FUZZ-IEEE)*, pages 1–7. IEEE, 2017.

[92] Florian Fenzl, Roland Rieke, and Andreas Dominik. In-vehicle detection of targeted can bus attacks. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pages 1–7, 2021.

[93] Said Ben Hassane Samir, Martin Raissa, Haifa Touati, Mohamed Hadded, and Hakim Ghazzai. Machine learning-based intrusion detection for securing in-vehicle can bus communication. *SN Computer Science*, 5(8):1082, 2024.

[94] Tien-Dat Le, Hoang Bao Huy Truong, Daehee Kim, et al. Multi-classification in-vehicle intrusion detection system using packet-and sequence-level characteristics from time-embedded transformer with autoencoder. *Knowledge-Based Systems*, 299:112091, 2024.

[95] Muhammad Sami, Matthew Ibarra, Anamaria C Esparza, Saleh Al-Jufout, Mehrdad Aliasgari, and Mohammad Mozumdar. Rapid, multi-vehicle and feed-forward neural network based intrusion detection system for controller area network bus. In *2020 IEEE Green Energy and Smart Systems Conference (IGESSC)*, pages 1–6. IEEE, 2020.

[96] Dogukan Aksu and Muhammed Ali Aydin. Mga-ids: Optimal feature subset selection for anomaly detection framework on in-vehicle networks-can bus based on genetic algorithm and intrusion detection approach. *Computers & Security*, 118:102717, 2022.

[97] Safa Boumiza and Rafik Braham. An anomaly detector for can bus networks in autonomous cars based on neural networks. In *2019 international conference on wireless and mobile computing, networking and communications (WiMob)*, pages 1–6. IEEE, 2019.

[98] Seunghyun Park and Jin-Young Choi. Hierarchical anomaly detection model for in-vehicle networks using machine learning algorithms. *Sensors*, 20(14):3934, 2020.

[99] Xing Zhang, Xiaotong Cui, Kefei Cheng, and Liang Zhang. A convolutional encoder network for intrusion detection in controller area networks. In *2020 16th International Conference on Computational Intelligence and Security (CIS)*, pages 366–369. IEEE, 2020.

[100] Omar Minawi, Jason Whelan, Abdulaziz Almehmadi, and Khalil El-Khatib. Machine learning-based intrusion detection system for controller area networks. In *Proceedings*

of the 10th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications, pages 41–47, 2020.

[101] Asma Alfardus and Danda B Rawat. Intrusion detection system for can bus in-vehicle network based on machine learning algorithms. In *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0944–0949. IEEE, 2021.

[102] Ahmed NasrEldin, Ayman M Bahaa-Eldin, and Mohamed A Sobh. In-vehicle intrusion detection based on deep learning attention technique. In *2021 16th International Conference on Computer Engineering and Systems (ICCES)*, pages 1–7. IEEE, 2021.

[103] Easa Alalwany and Imad Mahgoub. Classification of normal and malicious traffic based on an ensemble of machine learning for a vehicle can-network. *Sensors*, 22(23):9195, 2022.

[104] Defeng Ding, Lu Zhu, Jiaying Xie, and Jiaying Lin. In-vehicle network intrusion detection system based on bi-lstm. In *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP)*, pages 580–583. IEEE, 2022.

[105] Ch Ravi Kishore, D Chandrasekhar Rao, Janmenjoy Nayak, and HS Behera. Intelligent intrusion detection framework for anomaly-based can bus network using bidirectional long short-term memory. *Journal of The Institution of Engineers (India): Series B*, pages 1–24, 2024.

[106] Amit Chougule, Ishan Kulkarni, Tejasvi Alladi, Vinay Chamola, and Fei Richard Yu. Hybridsecnet: In-vehicle security on controller area networks through a hybrid two-step lstm-cnn model. *IEEE Transactions on Vehicular Technology*, 2024.

[107] Easa Alalwany and Imad Mahgoub. An effective ensemble learning-based real-time intrusion detection scheme for an in-vehicle network. *Electronics*, 13(5):919, 2024.

[108] Dheeraj Basavaraj and Shahab Tayeb. Towards a lightweight intrusion detection framework for in-vehicle networks. *Journal of Sensor and Actuator Networks*, 11(1):6, 2022.

[109] Kai Gao, Hao Huang, Linhong Liu, Ronghua Du, and Jinlai Zhang. A multi-attention based cnn-bilstm intrusion detection model for in-vehicle networks. In *2023 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 809–816. IEEE, 2023.

[110] Soner Can Kalkan and Ozgur Koray Sahingoz. In-vehicle intrusion detection system on controller area network with machine learning models. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6. IEEE, 2020.

[111] Wanting Gou, Haodi Zhang, and Ronghui Zhang. Multi-classification and tree-based ensemble network for the intrusion detection system in the internet of vehicles. *Sensors*, 23(21):8788, 2023.

[112] Haoyu Ma, Jianqiu Cao, Bo Mi, Darong Huang, Yang Liu, and Shaoqian Li. A gru-based lightweight system for can intrusion detection in real time. *Security and Communication Networks*, 2022(1):5827056, 2022.

[113] Muneeb Hassan Khan, Abdul Rehman Javed, Zafar Iqbal, Muhammad Asim, and Ali Ismail Awad. Divacan: Detecting in-vehicle intrusion attacks on a controller area network using ensemble learning. *Computers & Security*, 139:103712, 2024.

[114] Hsiao-Chung Lin, Ping Wang, Kuo-Ming Chao, Wen-Hui Lin, and Jia-Hong Chen. Using deep learning networks to identify cyber attacks on intrusion detection for in-vehicle networks. *Electronics*, 11(14):2180, 2022.

[115] Md Delwar Hossain, Hiroyuki Inoue, Hideya Ochiai, Doudou Fall, and Youki Kadobayashi. Long short-term memory-based intrusion detection system for in-

vehicle controller area network bus. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 10–17. IEEE, 2020.

[116] Mario Casillo, Simone Coppola, Massimo De Santo, Francesco Pascale, and Emanuele Santonicola. Embedded intrusion detection system for detecting attacks over can-bus. In *2019 4th International Conference on System Reliability and Safety (ICSRS)*, pages 136–141. IEEE, 2019.

[117] Mohd Nazeer, Areej Alasiry, Mohammed Qayyum, Vemana Karunakar Madhan, Gouri Patil, and Pulipati Srilatha. Enhancing cyber security in autonomous vehicles: A hybrid xg boost-deep learning approach for intrusion detection in the can bus. *Journal Européen des Systèmes Automatisés*, 57(5), 2024.

[118] Trieu Phong Nguyen, Heungwoo Nam, and Daehee Kim. Transformer-based attention network for in-vehicle intrusion detection. *IEEE Access*, 11:55389–55403, 2023.

[119] Miki E Verma, Michael D Iannacone, Robert A Bridges, Samuel C Hollifield, Bill Kay, and Frank L Combs. Road: The real ornl automotive dynamometer controller area network intrusion detection dataset (with a comprehensive can ids dataset survey & guide). *arXiv preprint arXiv:2012.14600*, 2020.

[120] Muhammad Sami. Intrusion detection in can bus, 2019.

[121] Adrian Taylor, Sylvain Leblanc, and Nathalie Japkowicz. Probing the limits of anomaly detectors for automobiles with a cyberattack framework. *IEEE Intelligent Systems*, 33(2):54–62, 2018.

[122] Mee Lan Han, Byung Il Kwak, and Huy Kang Kim. Anomaly intrusion detection method for vehicular networks based on survival analysis. *Vehicular communications*, 14:52–63, 2018.

[123] Guillaume Dupont, Alexios Lekidis, J. (Jerry) den Hartog, and S. (Sandro) Etalle. Automotive controller area network (can) bus intrusion dataset v2, 2019.

[124] Hacking and Countermeasure Research Lab (HCRL). In-vehicle network intrusion detection challenge, 2019.

[125] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. Network anomaly detection using lstm based autoencoder. In *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pages 37–45, 2020.

[126] Ashaar Alkhamaiseh, Mouhammd Alkasassbeh, and Jaafer Al-Saraireh. Unknown attack detection based on multistage one-class svm. In *2022 International Conference on Emerging Trends in Computing and Engineering Applications (ETCEA)*, pages 1–9, 2022.

[127] Baskoro Adi Pratomo, Pete Burnap, and George 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)*, pages 1–8. IEEE, 2018.

[128] Omid Avatefipour, Ameena Saad Al-Sumaiti, Ahmed M El-Sherbeeny, Emad Mahrous Awwad, Mohammed A Elmeligy, Mohamed A Mohamed, and Hafiz Malik. An intelligent secured framework for cyberattack detection in electric vehicles' can bus using machine learning. *Ieee Access*, 7:127580–127592, 2019.

[129] Sampath Rajapaksha, Harsha Kalutarage, M Omar Al-Kadri, Garikayi Madzudzo, and Andrei V Petrovski. Keep the moving vehicle secure: Context-aware intrusion detection system for in-vehicle can bus security. In *2022 14th International Conference on Cyber Conflict: Keep Moving!(CyCon)*, volume 700, pages 309–330. IEEE, 2022.

[130] Shashwat Khandelwal and Shanker Shreejith. Real-time zero-day intrusion detection system for automotive controller area network on fpgas. In *2023 IEEE 34th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 139–146. IEEE, 2023.

[131] Jake Guidry, Fahad Sohrab, Raju Gottumukkala, Satya Katragadda, and Moncef Gabbouj. One-class classification for intrusion detection on vehicular networks. In *2023 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1176–1182. IEEE, 2023.

[132] Prashanth Balaji and Majid Ghaderi. Neurocan: Contextual anomaly detection in controller area networks. In *2021 IEEE International Smart Cities Conference (ISC2)*, pages 1–7. IEEE, 2021.

[133] Heng Sun, Miaomiao Chen, Jian Weng, Zhiquan Liu, and Guanggang Geng. Anomaly detection for in-vehicle network using cnn-lstm with attention mechanism. *IEEE Transactions on Vehicular Technology*, 70(10):10880–10893, 2021.

[134] Pengcheng Wei, Bo Wang, Xiaojun Dai, Li Li, and Fangcheng He. A novel intrusion detection model for the can bus packet of in-vehicle network based on attention mechanism and autoencoder. *Digital Communications and Networks*, 2022.

[135] Pegah Mansourian, Ning Zhang, Arunita Jaekel, Mina Zamanirafe, and Marc Kneppers. Anomaly detection for connected autonomous vehicles using lstm and gaussian naïve bayes. In *International Conference on Wireless and Satellite Systems*, pages 31–43. Springer, 2023.

[136] Yilin Zhao, Yijie Xun, Jiajia Liu, and Siyu Ma. Gvids: A reliable vehicle intrusion detection system based on generative adversarial network. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 4310–4315. IEEE, 2022.

[137] Hongmao Qin, Mengru Yan, and Haojie Ji. Application of controller area network (can) bus anomaly detection based on time series prediction. *Vehicular Communications*, 27:100291, 2021.

[138] Izhar Ahmed Khan, Nour Moustafa, Dechang Pi, Waqas Haider, Bentian Li, and Alireza Jolfaei. An enhanced multi-stage deep learning framework for detecting malicious activities from autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):25469–25478, 2021.

[139] Victor Cobilean, Harindra S Mavikumbure, Chathurika S Wickramasinghe, Benny J Varghese, Timothy Pennington, and Milos Manic. Anomaly detection for in-vehicle communication using transformers. In *IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6. IEEE, 2023.

[140] Harini Narasimhan, Vinayakumar Ravi, and Nazeeruddin Mohammad. Unsupervised deep learning approach for in-vehicle intrusion detection system. *IEEE Consumer Electronics Magazine*, 12(1):103–108, 2021.

[141] Jie Wang and Xiuliang Mo. A can bus anomaly detection based on flxgboost algorithm. In *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, pages 1558–1564. IEEE, 2021.

[142] Kushagra Agrawal, Tejasvi Alladi, Ayush Agrawal, Vinay Chamola, and Abderrahim Benslimane. Novelads: A novel anomaly detection system for intra-vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):22596–22606, 2022.

[143] Jiahao Shi, Zhijun Xie, Li Dong, Xianliang Jiang, and Xing Jin. Ids-dec: A novel intrusion detection for can bus traffic based on deep embedded clustering. *Vehicular Communications*, 49:100830, 2024.

[144] Stefano Longari, Daniel Humberto Nova Valcarcel, Mattia Zago, Michele Carminati, and Stefano Zanero. Cannolo: An anomaly detection system based on lstm autoencoders for controller area network. *IEEE Transactions on Network and Service Management*, 18(2):1913–1924, 2020.

[145] Stefano Longari, Carlo Alberto Pozzoli, Alessandro Nichelini, Michele Carminati, and Stefano Zanero. Candito: improving payload-based detection of attacks on controller area networks. In *International Symposium on Cyber Security, Cryptology, and Machine Learning*, pages 135–150. Springer, 2023.

[146] Markus Hanselmann, Thilo Strauss, Katharina Dormann, and Holger Ulmer. Canet: An unsupervised intrusion detection system for high dimensional can bus data. *Ieee Access*, 8:58194–58205, 2020.

[147] Ch Ravi Kishore, D Chandrasekhar Rao, and HS Behera. Deep learning approach for anamoly detection in can bus network: An intelligent lstm-based intrusion detection system. In *International Conference on Computational Intelligence in Pattern Recognition*, pages 531–544. Springer, 2022.

[148] Sampath Rajapaksha, Harsha Kalutarage, M Omar Al-Kadri, Andrei Petrovski, and Garikayi Madzudzo. Beyond vanilla: Improved autoencoder-based ensemble in-vehicle intrusion detection system. *Journal of information security and applications*, 77:103570, 2023.

[149] Taeguen Kim, Jiyoon Kim, and Ilsun You. An anomaly detection method based on multiple lstm-autoencoder models for in-vehicle network. *Electronics*, 12(17):3543, 2023.

[150] Hyunjun Jo and Deok-Hwan Kim. Intrusion detection using transformer in con-troller area network. *IEEE Access*, 2024.

[151] Junchao Xiao, Hao Wu, and Xiangxue Li. Robust and self-evolving ids for in-vehicle network by enabling spatiotemporal information. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1390–1397. IEEE, 2019.

[152] Crash Reconstruction Research Consortium. Dodge can messages. https://www.engr.colostate.edu/~jdaily/tucrrc/DodgeCAN.html. (accessed 10 October 2024).

[153] Hyun Min Song and Huy Kang Kim. Discovering can specification using on-board diagnostics. *IEEE Design & Test*, 38(3):93–103, 2020.

[154] Mattia Zago, Stefano Longari, Andrea Tricarico, Michele Carminati, Manuel Gil Pérez, Gregorio Martínez Pérez, and Stefano Zanero. Recan–dataset for reverse engineering of controller area networks. *Data in brief*, 29:105149, 2020.

[155] Jiayan Zhang, Fei Li, Haoxi Zhang, Ruxiang Li, and Yalin Li. Intrusion detection system using deep learning for in-vehicle security. *Ad Hoc Networks*, 95:101974, 2019.

[156] Li Yang, Abdallah Moubayed, and Abdallah Shami. Mth-ids: A multitiered hybrid intrusion detection system for internet of vehicles. *IEEE Internet of Things Journal*, 9(1):616–632, 2022.

[157] Shu Nakamura, Koh Takeuchi, Hisashi Kashima, Takeshi Kishikawa, Takashi Ushio, Tomoyuki Haga, and Takamitsu Sasaki. In-vehicle network attack detection across vehicle models: A supervised-unsupervised hybrid approach. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1286–1291. IEEE, 2021.

[158] Auangkun Rangsikunpum, Sam Amiri, and Luciano Ost. Bids: An efficient intrusion detection system for in-vehicle networks using a two-stage binarised neural network on low-cost fpga. *Journal of Systems Architecture*, 156:103285, 2024.

[159] Mee Lan Han, Byung Il Kwak, and Huy Kang Kim. Event-triggered interval-based anomaly detection and attack identification methods for an in-vehicle network. *IEEE Transactions on Information Forensics and Security*, 16:2941–2956, 2021.

[160] Elies Gherbi, Blaise Hanczar, Jean-Christophe Janodet, and Witold Klaudel. Deep learning for in-vehicle intrusion detection system. In *Neural Information Processing: 27th International Conference, ICONIP 2020, Bangkok, Thailand, November 18–22, 2020, Proceedings, Part IV 27*, pages 50–58. Springer, 2020.

[161] Trieu-Phong Nguyen, Jeongho Cho, and Daehee Kim. Semi-supervised intrusion detection system for in-vehicle networks based on variational autoencoder and adversarial reinforcement learning. *Knowledge-Based Systems*, 304:112563, 2024.

[162] Jiaying Lin, Yehua Wei, Wenjia Li, and Jing Long. Intrusion detection system based on deep neural network and incremental learning for in-vehicle can networks. In *Inernational Conference on Ubiquitous Security*, pages 255–267. Springer, 2021.

[163] Geeta Kocher and Gulshan Kumar. Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges. *Soft Computing*, 25(15):9731–9763, 2021.

[164] Bo Li and Yevgeniy Vorobeychik. Feature cross-substitution in adversarial classification. *Advances in neural information processing systems*, 27, 2014.

[165] Fei Zhang, Patrick PK Chan, Battista Biggio, Daniel S Yeung, and Fabio Roli. Adversarial feature selection against evasion attacks. *IEEE transactions on cybernetics*, 46(3):766–777, 2015.

[166] Sk Tanzir Mehedi, Adnan Anwar, Ziaur Rahman, and Kawsar Ahmed. Deep transfer learning based intrusion detection system for electric vehicular networks. *Sensors*, 21(14):4736, 2021.

[167] Brooke Lampe and Weizhi Meng. A survey of deep learning-based intrusion detection in automotive applications. *Expert Systems with Applications*, 221:119771, 2023.

[168] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.

[169] Hamideh Taslimasa, S. Dadkhah, E. P. Neto, Pulei Xiong, Shahrear Iqbal, S. Ray, and A. Ghorbani. Imagefed: Practical privacy preserving intrusion detection system for in-vehicle can bus protocol. *IEEE BigDataSecurity*, 2023.

[170] Mirco Marchetti and Dario Stabili. Read: Reverse engineering of automotive data frames. *IEEE Transactions on Information Forensics and Security*, 14(4):1083–1097, 2018.

[171] Thien-Nu Hoang, Md Rezanur Islam, Kangbin Yim, and Daehee Kim. Canperfl: improve in-vehicle intrusion detection performance by sharing knowledge. *Applied Sciences*, 13(11):6369, 2023.

[172] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[173] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[174] Benson Mwangi, Tian Siva Tian, and Jair C Soares. A review of feature reduction techniques in neuroimaging. *Neuroinformatics*, 12:229–244, 2014.

[175] Jie Cai, Jiawei Luo, Shulin Wang, and Sheng Yang. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79, 2018.

[176] scikit-learn developers. Selectkbest. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html, 2025. Accessed: 2025-04-09.

[177] Ganjar Alfian, Muhammad Syafrudin, Imam Fahrurrozi, Norma Latif Fitriyani, Fransiskus Tatas Dwi Atmaji, Tri Widodo, Nurul Bahiyah, Filip Benes, and Jongtae Rhee. Predicting breast cancer from risk factors using svm and extra-trees-based feature selection method. *Computers*, 11(9):136, 2022.

[178] Dipti Theng and Kishor K Bhoyar. Feature selection techniques for machine learning: a survey of more than two decades of research. *Knowledge and Information Systems*, 66(3):1575–1637, 2024.

[179] Yi Li, Jing Lin, and Kaiqi Xiong. An adversarial attack defending system for securing in-vehicle networks. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2021.

[180] Acheme Samuel and Glory Nosawaru Edegbe. A systematic review of centralized and decentralized machine learning models: Security concerns, defenses and future directions. *NIPES-Journal of Science and Technology Research*, 6(4), 2024.

[181] Usman Shuaibu Musa, Megha Chhabra, Aniso Ali, and Mandeep Kaur. Intrusion detection system using machine learning techniques: A review. In *2020 international conference on smart electronics and communication (ICOSEC)*, pages 149–155. IEEE, 2020.

[182] Amal Hbaieb, Samiha Ayed, and Lamia Chaari. Federated learning based ids approach for the iov. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1–6, 2022.

[183] KM Faraoun and Aoued Boukelif. Neural networks learning improvement using the k-means clustering algorithm to detect network intrusions. *INFOCOMP Journal of Computer Science*, 5(3):28–36, 2006.

[184] Khattab M Ali Alheeti and Klaus McDonald-Maier. Intelligent intrusion detection in external communication systems for autonomous vehicles. *Systems Science & Control Engineering*, 6(1):48–56, 2018.

[185] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[186] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[187] Jack V Tu. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of clinical epidemiology*, 49(11):1225–1231, 1996.

[188] Guoqiang Peter Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462, 2000.

[189] Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5-6):352–359, 2002.

[190] Jianfa Wu, Dahao Peng, Zhuping Li, Li Zhao, and Huanzhang Ling. Network intrusion detection based on a general regression neural network optimized by an improved artificial immune algorithm. *PloS one*, 10(3):e0120976, 2015.

[191] Alex Shenfield, David Day, and Aladdin Ayesh. Intelligent intrusion detection systems using artificial neural networks. *Ict Express*, 4(2):95–99, 2018.

[192] Simon S Haykin et al. Neural networks and learning machines/simon haykin., 2009.

[193] Lulu Gao, Fei Li, Xiang Xu, and Yong Liu. Intrusion detection system using soeks and deep learning for in-vehicle security. *Cluster Computing*, 22(6):14721–14729, 2019.

[194] scikit-learn developers. Gridsearchcv — scikit-learn 1.6.1 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html, 2025. Accessed: 2025-04-09.

[195] Peiyu Xiong, Michael Tegegn, Jaskeerat Singh Sarin, Shubhraneel Pal, and Julia Rubin. It is all about data: A survey on the effects of data on adversarial robustness. *ACM Computing Surveys*, 56(7):1–41, 2024.

[196] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. *arXiv preprint arXiv:2102.01356*, 2021.

[197] Mohammad Y Abualhoul, Oyunchimeg Shagdar, and Fawzi Nashashibi. Visible light inter-vehicle communication for platooning of autonomous vehicles. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 508–513. IEEE, 2016.

[198] Abdallah Moubayed, Abdallah Shami, Parisa Heidari, Adel Larabi, and Richard Brunner. Edge-enabled v2x service placement for intelligent transportation systems. *IEEE Transactions on Mobile Computing*, 20(4):1380–1392, 2020.

[199] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[200] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[201] Flower AI. Flower a friendly federated learning framework, 2024.

[202] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[203] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978. IEEE, 2022.

[204] Jonathan Huang. Maximum likelihood estimation of dirichlet distribution parameters. *CMU Technique report*, 18, 2005.

[205] Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

[206] Judith Sáinz-Pardo Díaz and Álvaro López García. Study of the performance and scalability of federated learning for medical imaging with intermittent clients. *Neurocomputing*, 518:142–154, 2023.

[207] Kahou Tam, Li Li, Bo Han, Chengzhong Xu, and Huazhu Fu. Federated noisy client learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[208] Hong-You Chen, Cheng-Hao Tu, Ziwei Li, Han-Wei Shen, and Wei-Lun Chao. On the importance and applicability of pre-training for federated learning. *arXiv preprint arXiv:2206.11488*, 2022.

[209] Ishmeet Kaur and Adwaita Janardhan Jadhav. A comprehensive study on model initialization techniques ensuring efficient federated learning. *arXiv preprint arXiv:2311.02100*, 2023.

[210] Flower Framework. Use a federated learning strategy. Accessed: March 2024.

[211] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2011.

[212] Othmane Belarbi, Theodoros Spyridopoulos, Eirini Anthi, Ioannis Mavromatis, Pietro Carnelli, and Aftab Khan. Federated deep learning for intrusion detection in iot networks. In *GLOBECOM 2023-2023 IEEE Global Communications Conference*, pages 237–242. IEEE, 2023.

[213] Fred Lambert. Tesla is using new amd rdna 2 gpu in new model s/x, same as in playstation 5. https://electrek.co/2021/06/01/tesla-amd-rdna-2-gpu-new-model-s-x-same-playstation-5/, 2021. Accessed: 2024-11-18.

[214] Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 international joint conference on neural networks (IJCNN)*, pages 1–9. IEEE, 2020.

[215] Stefan Axelsson and David Sands. *Understanding intrusion detection through visualization*, volume 24. Springer Science & Business Media, 2006.

[216] Hampus Lundberg, Nishat I Mowla, Sarder Fakhrul Abedin, Kyi Thar, Aamir Mahmood, Mikael Gidlund, and Shahid Raza. Experimental analysis of trustworthy in-vehicle intrusion detection system using explainable artificial intelligence (xai). *IEEE Access*, 10:102831–102841, 2022.