Protecting vehicles from cyberattacks: context aware AI-based intrusion detection for vehicle CAN bus security.

RAJAPAKSHA, S.

2024

PROTECTING VEHICLES FROM CYBERATTACKS: CONTEXT AWARE AI-BASED INTRUSION DETECTION FOR VEHICLE CAN BUS SECURITY

Sampath Rajapaksha

A thesis submitted in partial fulfilment of the requirements of the
School of Computing
Robert Gordon University
for the degree of
Doctor of Philosophy

This research programme was carried out in collaboration with HORIBA MIRA Ltd.

August 2024

Abstract

Modern automobiles are equipped with a large number of Electronic Control Units (ECUs) which are interconnected through the Controller Area Network (CAN) bus for real-time data exchange. However, the CAN bus lacks security measures, rendering it susceptible to cyberattacks, endangering passenger safety. Although Artificial Intelligence (AI)-based Intrusion Detection Systems (IDSs) can detect these attacks, achieving higher detection rates in near real-time poses challenges. This research aims to enhance In-vehicle Networks (IVN) attack detection by developing a deployable AI-based IDS.

First, A lightweight context-aware IDS named CAN-CID is introduced, employing a combination of a Gated Recurrent Unit (GRU)-based Recurrent Neural Network (RNN) model and a time-based model. CAN-CID is designed to detect injection and masquerade attacks on the CAN bus. It achieved an F1 score of over 99% on three publicly available CAN attack datasets for 10 injections and three masquerade attacks, outperforming baseline models. To overcome the challenge of requiring a large dataset for effective attack detection with the GRU-based model for medium, and low frequent IDs, CAN-ODTL, a novel on-device transfer learning technique, is introduced. CAN-ODTL outperformed the pre-trained and baseline models with over 99% detection rate for realistic attacks. CAN-ODTL outperformed pre-trained and baseline models with a detection rate exceeding 99% for realistic attacks. CAN-ODTL is designed to be trained with a larger dataset compared to CAN-CID model to learn the majority of benign patterns of medium and low-frequency IDs, thus enhancing its ability to detect attacks targeting such IDs. As streaming learning approaches such as CAN-ODTL are susceptible to data poisoning attacks, an anomaly detection method leveraging the Mahalanobis distance is employed to identify and eliminate poisoned data samples before model retraining. Evaluation on a real dataset with varying percentages of data poisoning attacks demonstrates the method's high accuracy of 100% in detecting poisoned samples. While CAN ID-based CAN-ODTL is effective against injection and certain masquerade attacks, it faces challenges in detecting attacks that only alter the payload field. To address this limitation, an improved Autoencoder (AE)-based model, known as Latent AE is introduced for detecting attacks aimed at the payload data. The ensemble of the GRU-based RNN model and Latent AE demonstrated its superiority over baseline models, exhibiting near real-time detection latency.

In response to the current lack of realistic attack datasets, a novel CAN bus dataset is

presented. The improved models of proposed CAN-ODTL and Latent AE models are then deployed in a real vehicle and evaluated with real-world attacks. This demonstrated the effectiveness of the proposed IDS by achieving over a 99% attack detection rate for 23 attacks with near-real time detection latency of 25ms. These results highlight the effectiveness of employing multiple IDSs, each utilizing distinct fields of the CAN data, in detecting attacks and achieving near-real-time detection.

Keywords: Controller Area Network (CAN), Automotive cybersecurity, In-vehicle network attacks, Anomaly detection, Intrusion Detection System (IDS)

Declaration of Authorship

I declare that I am the sole author of this thesis and that all verbatim extracts contained in the thesis have been identified as such and all sources of information have been specifically acknowledged in the bibliography.

Acknowledgements

I extend my deepest gratitude to my supervisory team, Dr. Harsha Kalutarage, Dr. Omar Al-kadri, Dr. Andrei Petrovski, Dr. Garikayi Madzudzo and Dr. Madeline Cheah for their unwavering support, valuable guidance, and insightful advice throughout my research journey. Their critical and robust supervision played a pivotal role in the success of my work, and I am profoundly thankful for their contributions.

Special thanks must go to my principal supervisor Dr. Harsha Kalutarage for his invaluable supervision, insightful guidance, continuous support, and motivation. His persistent help has been a driving force, and I am sincerely grateful for his mentorship.

I would also like to express my heartfelt appreciation to my entire family, including my father, mother, wife, sisters, and friends, for their unwavering support and encouragement. Their belief in me has been a constant source of strength. Additionally, I am thankful to the administrative and IT teams at the School of Computing for their support in facilitating my research endeavours. Special thanks to Horiba MIRA Ltd for partially funding this research project, as well as for their assistance with data collection and model deployment.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

The growth of information technologies has driven the development of the transportation sector, including connected and autonomous vehicles. Consequently, modern automobiles are equipped with a large number of ECUs to deliver services like adaptive cruise control, lane departure warning, automated parking assistance, and infotainment systems. These features aim to ensure safety, offer driver assistance, and enhance the overall comfort of passengers. To meet the demands of these functionalities, vehicle networks must support near real-time data transmission, providing ample bandwidth and ensuring reliable communication. The CAN bus serves as a pivotal technology, facilitating near real-time data transmission between ECUs with the required reliability for effective in-vehicle communication.

1.1 Research Motivation

Modern automobiles are becoming intelligent, complex, and highly connected. In 1980, vehicles had only 1% of electronic equipment compared to their mechanical counterparts. However, nowadays, electronic components have increased up to 50% [3]. This will continue to increase with the advent of autonomous cars, which will rely on powerful computer systems, a range of sensors, networking, and satellite navigation, all of which require electronics. Furthermore, modern vehicles embody software that exceeds 100 million lines of code, and it is expected to grow beyond 300 million lines of code in the near future [4]. Software on modern automobiles run on 70-100 microprocessor-based ECUs that are networked throughout the vehicle [4]. In addition to the multitude of ECUs, modern vehicles are equipped with multiple sensors, actuators, cameras, radars,

and communication devices [5, 6]. These systems aim to enhance performance, efficiency, intelligent services, and safety for automobile users by collecting and interpreting diverse data [5]. However, simultaneously, these systems contribute to making modern automobiles significantly more complex.

The seamless exchange of diverse information between automotive systems necessitates a unified network capable of real-time data transmission with adequate bandwidth and reliability [7]. The CAN bus fulfils these requirements and stands out as the most widely used in-vehicle network protocol. However, the CAN bus has well-known security flaws, such as a lack of authentication, broadcast transmission, lack of encryption, and an ID-based priority mechanism. Moreover, owing to the increasing complexity of modern automobiles, the CAN bus is becoming more exposed to the external world through interfaces like the on-board diagnostics II (OBD-II) port and various wireless communication channels such as WiFi, Bluetooth, radio systems, and telematics units [8, 9]. Exploiting CAN bus vulnerabilities and utilising diverse attack vectors, security researchers have demonstrated the potential for attacks against various vehicle brands [10, 9, 11]. These attacks empower malicious actors to seize physical control of the vehicle and activate different functions, posing a direct threat to the safety of drivers, passengers, and the surrounding environment. In response to these risks, various regulatory bodies have defined regulations and standards for the cyber security of vehicles. The Global Forum for Harmonization of Vehicle Regulations of the United Nations Economic Commission for Europe (UNECE) introduced two cybersecurity regulations in 2021: UNECE R155 [12] and UNECE R156 [13]. UNECE R155 is primarily focused on providing uniform provisions for vehicle cybersecurity and the Cyber Security Management System (CSMS). CSMS represents a systematic risk-based approach defining organisational processes, responsibilities and governance to treat risk associated with cyber threats to vehicles and protect them from cyberattacks. This regulation mandates vehicle manufacturers to undertake essential measures such as implementing appropriate cybersecurity measures in the design of the vehicle type, detecting and responding to possible cyberattacks and logging data to support the detection of cyberattacks and providing data forensic capability to enable analysis of attempted or successful cyberattacks. In contrast, UNECE R156 focused on uniform provisions for software updates and software update management system. Additionally, ISO/SAE 21431:2021 outlines cybersecurity engineering requirements for road vehicles. This standard specifies the engineering protocols necessary for cybersecurity risk management across various stages, including concept development,

product development, production, operation, maintenance, and decommissioning of electrical and electronic (E/E) systems within road vehicles, encompassing their components and interfaces [14].

With the emergence of regulations, standards, and the increased threat of cyberattacks, automotive manufacturers are proactively exploring security measures to protect vehicles from potential cyber threats [15, 16].

1.2 Research Questions and Objectives

IDSs have been proposed to identify cyberattacks in IVNs [17]. Embedded system-based IDSs enhance data safety and privacy compared to cloud-based IDSs [18]. However, there are constraints associated with embedded system-based IDSs for IVNs, including limitations in memory storage, computational power, bandwidth, and energy consumption [19]. For example, the Raspberry Pi 4B, an edge computing device suitable for deploying IDSs, features a four-core ARM Cortex A72, 1.5GHz CPU with 8GB of memory and a power consumption of 5-10W. In contrast, the Nvidia Jetson AGX Xavier, another popular device, includes an eight-core ARM v8.2, 2.26GHz CPU with 32GB of memory and requires 10-30W of power. Additionally, costs related to hardware, software development, updates, and the economic viability of the IDS must be considered. For instance, while the Nvidia Jetson AGX Xavier offers improved computational power compared to the Raspberry Pi 4B, it is significantly more expensive (£600) compared to the Raspberry Pi 4B (£75). Beyond these constraints, challenges such as the scarcity of realistic attack data, the unavailability of CAN data specifications, and the necessity for real-time detection capabilities further complicate the development of IDS for IVNs. Considering these constraints, to address these challenges, this thesis explores the following research questions (RQ):

RQ1: What are the common attack types and behaviours observed in IVN systems, the distinctive features within CAN bus data that can be utilized for effective cyberattack detection, and what are the existing solutions and their limitations? Chapter 3 addresses this RQ.

RQ2: How can benign CAN-ID sequences be employed for near real-time attack detection and enhance the accuracy of attack detection? Initial literature review suggests that while benign CAN-ID sequences can aid in detecting attacks, existing models are susceptible to false positives. Chapter 5 and Chapter 6 address this RQ.

RQ3: How can the CAN payload field be leveraged to detect sophisticated IVN attacks without prior knowledge of CAN specifications? The initial literature review suggests that developing a lightweight CAN payload-based IDS is challenging without prior knowledge of CAN specifications. Chapter 7 addresses this RQ.

RQ4: How effective are the developed IDSs when tested on a real vehicle against different cyberattacks? Chapter 9 addresses this RQ.

The aim of this research is to enhance IVN security through the development of a practically deployable AI-based IDS capable of detecting cyberattacks on the automotive CAN bus. To attain this aim and address the aforementioned research questions, five key research objectives (RO) have been outlined in this thesis as follows:

RO1: Conduct a thorough examination of existing solutions, their limitations, and IVN attacks, aiming to identify potential AI-based countermeasures for enhancing IVN security. This involves studying proposed AI-based solutions, categorising attack types, analysing attack behaviours, defining distinctive features within CAN bus data, and exploring feature engineering techniques to effectively detect cyberattacks in the CAN bus (RQ1).

RO2: Develop a lightweight AI-based IDS utilizing benign CAN ID sequences to improve attack detection capabilities while prioritizing secure model training (RQ2). Achieving this objective will result in a deployable IDS that does not necessitate attack data during model training.

RO3: Derive significant features from CAN payload fields without prior knowledge of CAN specifications, leading to the development of a lightweight IDS specifically designed to detect sophisticated attacks (RQ3). Realising this objective will improve the sophisticated attack detection and generalization capability of the IDS.

RO4: Generate a comprehensive CAN bus attack dataset by collecting data from a moving vehicle, providing a valuable resource for evaluating IDS performance (RQ2, RQ3). The preliminary literature review underscores the necessity for such a dataset collected under realistic driving conditions.

RO5: Deploy the IDSs developed in RO2 and RO3 on a real vehicle, conducting a comprehensive evaluation that includes injection, suspension, and masquerade attacks to assess their effectiveness in a practical setting (RQ4). Achieving this goal will assist in tackling the real-world challenges associated with CAN bus attack detection.

1.3 Scope of the Research

Given data availability, this research focuses solely on CAN bus security and does not address the security of other IVN protocols such as FlexRay, Local Interconnect Network (LIN), Media Oriented System Transport (MOST), or Vehicular Ad-hoc Networks (VANET). Lightweight AI-based algorithms are employed to detect cyberattacks with an emphasis on on-device deployment, thus excluding cloud-based IDS deployments. Due to time and resource constraints, the deployment experiments will be conducted using only one vehicle. The AI-based algorithms are developed with aforementioned constraints in mind, specifically for embedding devices and addressing IDS development challenges. This research focuses solely on detecting attacks on the CAN bus and does not cover attack prevention techniques. The research questions and objectives are selected in alignment with the defined scope of the study.

1.4 Contributions

This thesis significantly advances the current state of knowledge by creating a practically deployable IDS for IVN security. The following list provides a concise summary of the primary contributions made by this research:

- The first contribution of this thesis involves implementing a lightweight IDS based on CAN IDs. This context-aware IDS is designed to effectively detect a diverse range of IVN attacks while exhibiting near real-time detection capabilities.
- The second contribution of this thesis revolves around enhancing the developed CAN ID-based IDS by incorporating on-device transfer learning. This improvement enables the continuous enhancement of attack detection capabilities through the utilization of streaming CAN data. Furthermore, the thesis introduces an effective technique to counteract data poisoning attacks specifically targeted at the CAN IDS on-device transfer learning with streaming data.
- The third contribution of this thesis involves the proposal of an enhanced AE-based IDS designed to detect cyberattacks that solely manipulate the CAN payload field. The introduced AE model addresses the challenge of overgeneralization encountered by vanilla AEs when applied to anomaly detection in the CAN bus. Furthermore, a novel feature selection technique is introduced to mitigate data complexity by eliminating weakly associated features, thereby enhancing attack detection capabilities.

- The fourth contribution of this thesis is the creation of a CAN bus attack dataset captured during real-world driving conditions. This introduces CAN-MIRGU, a novel and publicly available dataset of CAN bus attacks obtained from a modern vehicle equipped with autonomous driving capabilities, operating under real-world driving conditions. The dataset encompasses physically verified attacks, addressing the existing gap in publicly available datasets by featuring realistic attacks within dynamic driving environments.

- The fifth contribution of this thesis involves deploying the developed IDSs into a real vehicle by integration into a resource-constrained Raspberry Pi device. This deployment further optimizes the developed CAN ID and payload-based IDSs to overcome practical challenges encountered in real-world settings.

1.4.1 List of Publications

Following is the list of publications produced during the work presented in this thesis:

Journal Publications:

- Rajapaksha, S., Kalutarage, H., Al-Kadri, M. O., Petrovski, A., & Madzudzo, G. (2023). Beyond vanilla: Improved autoencoder-based ensemble in-vehicle intrusion detection system. Journal of Information Security and Applications, 77, 103570.

- Rajapaksha, S., Kalutarage, H., Al-Kadri, M. O., Petrovski, A., Madzudzo, G., & Cheah, M. (2023). Ai-based intrusion detection systems for in-vehicle networks: A survey. ACM Computing Surveys, 55(11), 1-40.

Conference Publications:

- Rajapaksha, S., Kalutarage, H., Petrovski, A., Madzudzo, G., & Al-Kadri, M. O. StreamShield: Preventing Data Poisoning Attacks in In-Vehicle Intrusion Detection Training with Streaming Data. In 37th IEEE Computer Security Foundations Symposium 2024 (Under review)

- Rajapaksha, S., Madzudzo, G., Kalutarage, H., Petrovski, A., & Al-Kadri, M. O. (2024, February). CAN-MIRGU: A Comprehensive CAN Bus Attack Dataset from Moving Vehicles for Intrusion Detection System Evaluation. In Symposium on Vehicles Security and Privacy. Internet Society.

- Rajapaksha, S., Kalutarage, H., Al-Kadri, M. O., Petrovski, A., & Madzudzo,

G. (2023, February). Improving in-vehicle networks intrusion detection using on-device transfer learning. In Symposium on vehicles security and privacy (Vol. 10).

- Rajapaksha, S., Kalutarage, H., Al-Kadri, M. O., Madzudzo, G., & Petrovski, A. V. (2022, May). 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) (Vol. 700, pp. 309-330). IEEE.

1.5 Thesis Structure

The reminder of the thesis is structured as follows.

Chapter 2 provides the necessary background for readers, aiding their comprehension of the subsequent chapters in the thesis. This chapter delves into the characteristics of IVN, with a specific focus on the CAN bus, and explores common attacks that target the IVNs. Additionally, it provides the background for AI-based techniques employed in this research.

Chapter 3 offers an extensive literature review of AI-based IDSs designed to identify cyberattacks on IVNs. The works are systematically categorized using a unique taxonomy that emphasizes the utilization of CAN data fields to detect diverse attacks and the application of AI-based techniques. Additionally, the chapter explores the resilience of AI models, benchmark datasets, and highlights research gaps and future directions that serve as inspiration for contributions in the subsequent chapters.

Chapter 4 outlines the methodology employed in this thesis. This is explained using a process diagram that illustrates the interconnections among each chapter.

Chapter 5 introduces CAN-CID, a lightweight context-aware CAN IDS leveraging the CAN ID field. This model adopts an ensemble approach, combining a GRU network with a time-based model to enhance the effectiveness of attack detection.

Chapter 6 enhances the GRU-based model employed in CAN-CID in Chapter 5 by incorporating the on-device transfer learning technique for retraining the IDS with streaming CAN data. Recognizing the susceptibility of this approach to data poisoning attacks, the chapter introduces an effective technique to prevent such attacks during the retraining process.

Chapter 7 presents a novel AE-based IDS designed to detect sophisticated masquerade

attacks that alter only the characteristics of the CAN payload data. This approach assumes no prior knowledge of the CAN data specification and extracts only the essential features. A novel feature reduction technique is employed to enhance the model's lightweight nature and improve the effectiveness of attack detection.

Chapter 8 introduces CAN-MIRGU, an extensive CAN bus attack dataset collected from moving vehicles for the purpose of IDS evaluation. This chapter delves into a detailed discussion of attacking a moving vehicle and analyzes both benign and attack data, contributing to the advancement of CAN IDS research.

Chapter 9 details the deployment of IDS in a real vehicle by integration into a resource-constrained edge device. This chapter discusses the enhancements implemented for CAN ID and payload-based models, introduced in Chpater 6 and Chapter 7, to address the challenges associated with near-real-time detection.

Chapter 10 concludes this thesis. It provides a summary of the contributions and key outcomes of this research. Additionally, it discusses future research directions to highlight potential areas for improvement.

Chapter 2

Research Background

Modern vehicles incorporate numerous microprocessor-based ECUs, such as engine and power train control. Efficient communication between these ECUs necessitates a unified network capable of real-time data transmission with ample bandwidth and reliability. The CAN bus protocol fulfils these requirements, emerging as the most prevalent in-vehicle network protocol. Despite CAN's effective communication capabilities, it lacks essential security features like message encryption and authentication. These vulnerabilities render the CAN bus susceptible to cyberattacks. This chapter aims to provide a comprehensive background on in-vehicle networks, focusing specifically on the CAN Bus and highlighting its vulnerabilities, which make it susceptible to cyberattacks such as injection, masquerade, and suspension.

2.1 In-vehicle Networks (IVNs)

Modern automobiles are equipped with a variety of sensors, actuators, cameras, radars and communication devices [5, 6]. The primary objective of these systems is to improve performance, efficiency, intelligent services, and user safety. This is achieved by collecting, interpreting, and communicating diverse data both within these systems and with other vehicles and the surrounding infrastructure. Vehicle networks play a crucial role in facilitating communication for these automotive systems, which can be categorised into two main types: external and internal networks. The internal network is also referred to as in-vehicle or intra-vehicle network. The external network can be further classified as Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I), collectively known as VANETs. The term Vehicle-to-Everything (V2X) is sometimes used to encompass both

V2V and V2I [3]. This thesis focuses on IVNs, particularly the CAN bus.

In addition to their designated functions, ECUs engage in communication with other ECUs through various standard in-vehicle communication protocols found in automobiles. These protocols include the CAN bus, FlexRay, LIN and MOST [20]. The CAN bus is used for critical vehicle systems such as engine management, transmission and anti-brake systems. FlexRay, introduced in 2000 by the FlexRay consortium, is a time-triggered in-vehicle communication protocol that is used for high-end applications inside vehicles such as power train and safety functions [1]. It offers higher bandwidth and enhanced fault tolerance capabilities with a maximum baud rate of 10Mbps and a payload length of 254 bytes. Despite its advantages, FlexRay is more expensive than the CAN network and is more susceptible to Denial of Service (DoS) and spoofing attacks [21]. Media Oriented Systems Transport (MOST) serves as an In-vehicle network designed specifically for transmitting multimedia data. It operates at bandwidths of 25, 50, and 150Mbps, making it well-suited for multimedia applications. On the other hand, Local Interconnect Network (LIN) is a cost-effective in-vehicle protocol commonly employed in less critical applications such as seat belts, door locks, mirrors, batteries, and temperature monitoring.

The in-vehicle network system is divided into four domains based on its functionality and whether it necessitates real-time data [1, 22]. These domains encompass the power train, chassis, body, and telematics and infotainment domains. These domains exhibit distinctions in the functions they deliver and the network performance and quality they demand. Consequently, each domain has unique performance and response time requirements. All domains are interconnected through a gateway control unit, as shown in Figure 2.1.

- Powertrain Domain: This domain holds paramount significance as it demands real-time responsiveness. Consequently, it is interfaced with a High-Speed CAN bus or CAN Flexible Data (CAN FD), which extends the capabilities of the standard CAN bus protocol. Within this domain reside essential ECUs responsible for managing critical components like transmission and engine speed.

- Chassis domain: Comprising ECUs such as brake control and suspension control units, this domain offers real-time and safety-critical functions within the vehicle. Consequently, it is connected to CAN High, CAN FD, or FlexRay networks.

- Body Domain: Within this domain, the ECUs tasked with managing various functions like dashboard displays, windshield wipers, lighting systems, power windows,

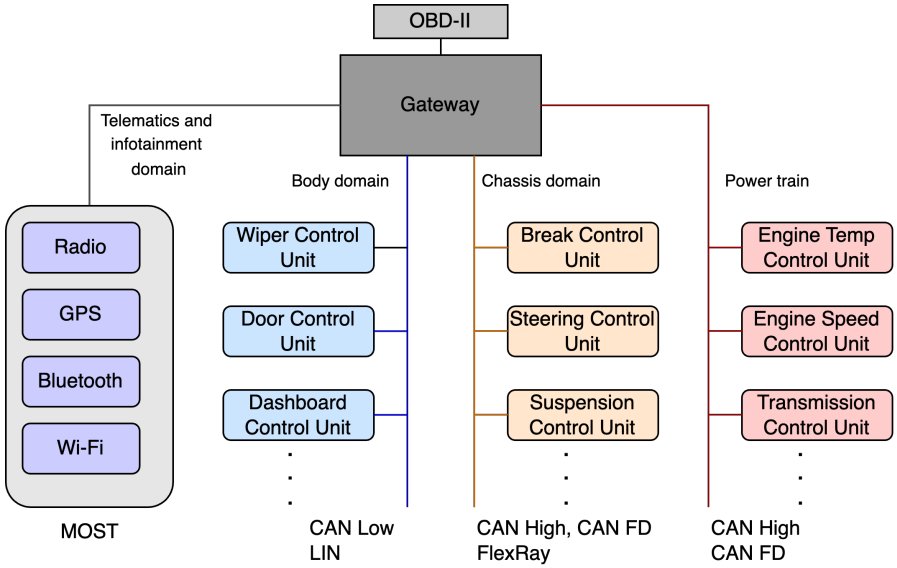


Figure 2.1: In-vehicle network system topology, adapted from [1].

and seat adjustments. Given that these operations typically do not necessitate immediate reactions, they are typically connected to either CAN Low or LIN networks.

- Telematics and Infotainment Domain: This domain oversees the management of various communication, information, and entertainment services within a vehicle. It is responsible for functions such as in-car navigation, CD and DVD players, rear-seat entertainment systems, driving assistance features, and wireless interfaces. It is typically connected to the MOST network.

2.2 Electronic Control Units (ECUs)

An ECU typically comprises sensors, actuators, a control unit, a CAN module, and a transceiver, as depicted in Figure 2.2. Essential components of the control unit include a microcontroller with input and output memories, as well as a program memory. During ECU operation, sensor data is received and stored in the input memory in a chronological order. Subsequently, the microcontroller processes these input values based on the configured programme, with the results stored in respective output memories. These outcomes are then transmitted to the actuators for action. Each control unit manages CAN messages using a dedicated CAN memory area for received and sent messages. The CAN module oversees the data transfer process for CAN messages, with separate sections for

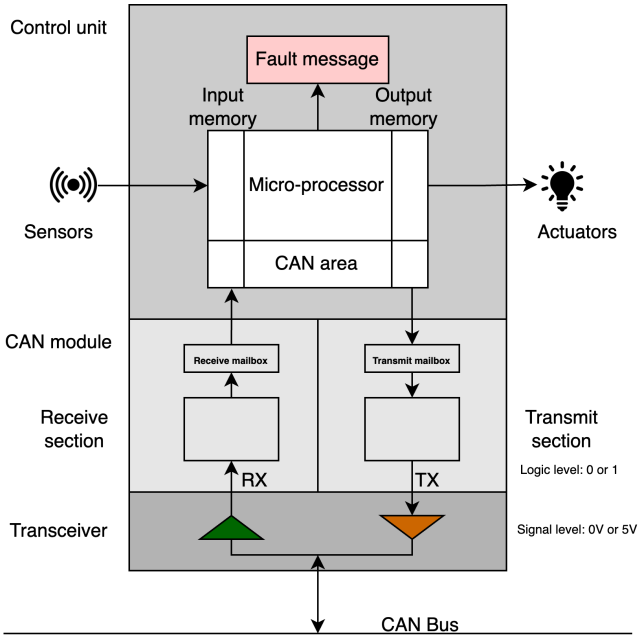


Figure 2.2: Functional units of an ECU, adapted from [2].

receiving and sending to handle incoming and outgoing messages. The transceiver serves as a transmitter and receiver amplifier, converting the serial bit stream (logic level) from the CAN module into electrical voltage values (line level) and vice versa. These voltage values facilitate transmission over the copper wires of the CAN bus. The transceiver is connected to the CAN module via either the TX line (transmit line) or the RX line (receive line). The RX line is directly linked to the CAN bus whereas the TX line is normally connected via an open collector allowing continuous monitoring of bus signals.

2.3 Controller Area Network (CAN Bus)

Among various network protocols, the CAN bus stands out as the predominant choice for in-vehicle communication, owing to its numerous advantages such as low cost, high speed, lightweight design, robustness, and simplified installation [1, 23]. The CAN bus operates as a message-based protocol, facilitating communication between different ECUs [19]. The CAN bus is further categorised into the High-Speed CAN bus and Low-Speed CAN bus based on data rates. The high-speed variant operates with a bit rate ranging from 125Kbps to 1Mbps, while the low-speed variant ranges from 5kbps to 125kbps. The CAN bus supports a payload of up to 8 bytes. ECUs with time-critical functions, such as engine

control and transmission control, are typically connected to the high-speed CAN bus, while ECUs handling less time-sensitive functions, such as door control and light control, are connected to the low-speed CAN bus. These two buses are interconnected through a gateway [24]. Additionally, the CAN Flexible Data (CAN-FD) protocol extends the capabilities of the traditional CAN bus by supporting a bit rate of up to 8Mbps and a maximum payload of 64 bytes [25].

Despite the advantages offered by the CAN bus, it is vulnerable to cyberattacks due to various vulnerabilities such as [17]:

- **Lack of authentication:** The absence of authentication in the CAN bus allows any ECU to transmit a frame with a CAN ID belonging to another ECU. Exploiting a compromised ECU, an attacker can inject malicious frames.
- **Broadcast domain:** The CAN bus functions as a broadcast domain, meaning that all nodes receive CAN frames transmitted through the network. This characteristic renders the CAN bus susceptible to sniffing attacks, enabling an attacker to eavesdrop on all messages.
- **Absence of encryption:** Due to time constraints, CAN messages are not encrypted, making it easy for cyber attackers to collect and analyze these messages.
- **ID-based priority:** The CAN network employs ID-based priority to manage multiple concurrent messages, where lower IDs indicate higher priority. Malicious nodes can continuously transmit frames with lower IDs, initiating a DoS attack.

2.3.1 CAN Bus Data Transmission Process

Sensors are responsible for detecting physical values, such as engine speed, which are initially represented as binary values. These binary values are then converted into a serial bit stream, as illustrated in Figure 2.3. The resulting bit stream is transmitted over the TX line (transmit line) to the transceiver (amplifier). The transceiver further converts the bit stream into voltage values, which are sequentially transmitted over the bus line. During reception, the transceiver reverses this process by converting incoming voltage values back into a bit stream, which is then sent over the RX line (receive line) to the control units. Subsequently, the control units decode the bit stream, converting the serial binary values back into meaningful messages. Actuators, based on the received messages, then execute the appropriate actions, such as adjusting the speedometer to reflect the current speed of the vehicle. Regularly, sensor readings are stored in the input

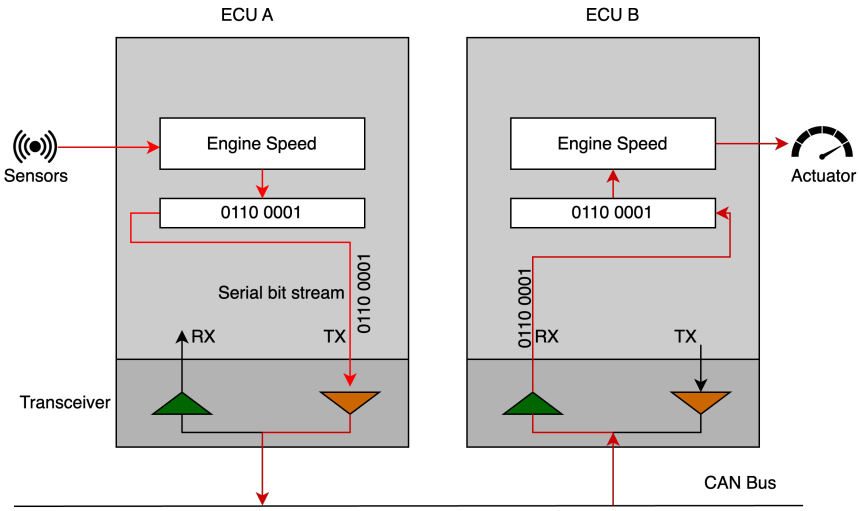


Figure 2.3: CAN bus data transmission, adapted from [2].

memory of ECU microcontrollers. Consequently, for all non-diagnostic messages, ECUs typically transmit messages at a fixed interval [9]. This recurrent behaviour establishes a sequential pattern for CAN ID sequences [26].

2.3.2 CAN Bus Data Frame

A CAN frame has a specific message structure defined in a database-like file known as DataBase CAN (DBC) file. This is a confidential proprietary of vehicle manufacture and contains all the necessary information of a specific vehicle related to ECUs, CAN messages, signals, message IDs, message frequency, and payload of the CAN frame [27]. There are four CAN frame types that can be identified as data frame, remote frame, overload frame, and error frame [23]. These frames exhibit varying lengths and serve distinct roles within the CAN communication system. However, it is noteworthy that, among these frames, only the data frames contribute significantly to the operations in the CAN communication process [28]. CAN data frame consists of seven fields that support data transmission from the transmitter to the receiver (ECUs). Figure 2.4 illustrates the fields of a CAN data frame with respective sizes. The seven fields of the CAN data frame are described below.

- **Start of Frame (SOF):** The start of frame specifies the beginning of a CAN frame. It uses the dominant bit (logical 0) to inform the beginning of CAN frame transmission to other nodes.

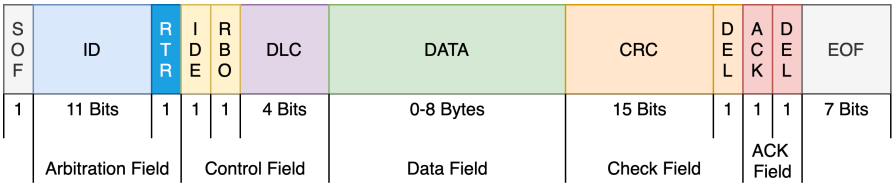


Figure 2.4: CAN bus data frame

- **Arbitration field (CAN-ID):** The Arbitration field, also known as the arbitration ID or simply ID, prioritizes messages when multiple ECUs concurrently transfer messages. For instance, in a scenario where two ECUs with CAN IDs 0x0D0 (000011010000 in binary) and 0x2E1 (001011100001 in binary) attempt to transmit messages simultaneously, the ECU with ID 0x0D0 gains bus access due to its lower value (higher priority). Typically, the CAN ID is 11 bits, and in the extended format, it extends to 29 bits. The Remote Transmission Request (RTR) distinguishes between data frames and remote frames. Each ECU is typically assigned one or more IDs, ensuring uniqueness for each ECU.

- **Control field (DLC):** The Control field is a 6-bit section that includes the Data Length Code (DLC) comprising 4 bits, which identifies the payload's length, and two additional bits reserved for future use.

- **Data field:** The Data field contains the actual information intended for transmission on the CAN bus, also known as the payload of the CAN frame. The payload can range from 0 to 8 bytes and may include sensor, category, constant, or cyclical counter data [29].

- **CRC field:** The Cyclic Redundancy Code (CRC) field, also referred to as the safety field, consists of 15 bits followed by a 1-bit delimiter. Its purpose is to verify the validity of the frame.

- **Acknowledge field (ACK):** The Acknowledge field, also known as the confirmation field, comprises a 1-bit acknowledgment and a 1-bit delimiter. This field ensures that the receiver nodes acknowledge the reception of the CAN frames.

- **End of frame (EOF):** The End of Frame specifies the end of the CAN frame.

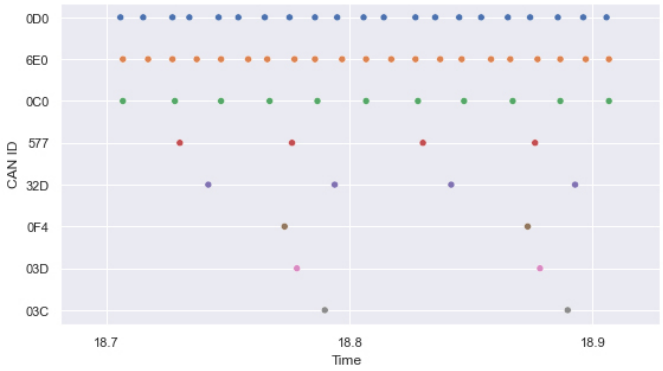


Figure 2.5: Frame transmission of selected IDs

2.3.3 CAN Bus Data Analysis

As aforementioned, during normal driving conditions, the majority of CAN frames transmit based on predefined frequencies, resulting in a sequential pattern for CAN IDs. As illustrated in Figure 2.6, this transmission frequency can vary from around 100 frames per second to only a few frames per second, depending on the CAN IDs. The sequential behaviour of normal CAN ID frames is illustrated in Figure 2.5 , which shows the ID transmission for a few selected IDs from a sample of the CAN dataset in the Real ORNL Automotive Dynamometer (ROAD) CAN intrusion dataset [30]. Our analysis of other publicly available CAN bus datasets also exhibits this behaviour regardless of the vehicle's make, model, or year. Additionally, high-priority IDs transmit frames more frequently, while low-priority IDs transmit frames less frequently. As a result, within a unit of time, frames with certain IDs transmit approximately the same number of frames. An analysis using a sample of the ROAD dataset with selected IDs for a one-second period clearly shows this behaviour in Figure 2.6. This behaviour can also be observed in other publicly available CAN bus datasets. The payload field includes different data types, such as sensor readings and categorical values, which change with the vehicle's driving behaviour. For instance, sensor readings related to vehicle speed might exhibit a time series behaviour within a fixed continuous value range, such as 0 to 200. Conversely, data values related to gear changes might display an ordinal categorical behaviour within a range of 0 to 5. All these payload data specifications are defined in the DBC file. These properties in the ID field and the payload field can be utilised to detect anomalous behaviours in CAN data transmission.

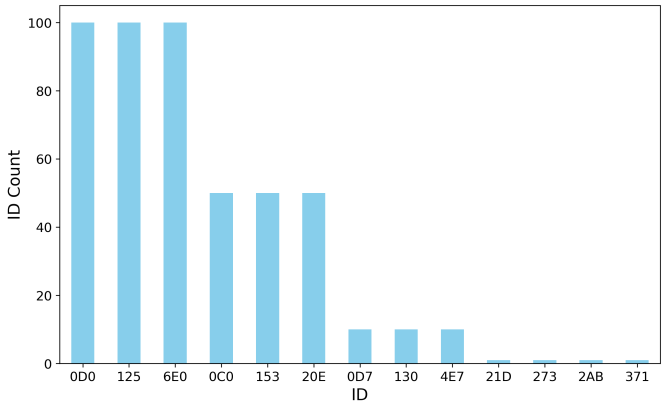


Figure 2.6: Number of frames transmitted for selected IDs within a one-second period

2.4 Attacks on CAN Bus

Due to the absence of authentication, broadcast domain vulnerability, lack of encryption, and reliance on ID-based priority, the CAN bus is susceptible to various types of cyberattacks. To execute these attacks, an attacker needs to gain access to the CAN bus. In practice, this can be achieved by connecting an OBD-II dongle while the vehicle is parked or by obtaining remote access [9]. Remote attack surfaces include the anti-theft system, tire pressure monitoring system (TPMS), remote keyless entry, Bluetooth, radio systems, Wi-Fi, cellular networks and telematics units [8, 9]. Various previous experimental research studies have demonstrated the feasibility of these attacks in real-world conditions [9, 31, 8, 11]. Table 2.1 presents some of the experimental attacks that were carried out on IVNs. Attacks on the CAN bus can be mainly classified into three categories as injection (fabrication), suspension and masquerade (impersonation) attacks [32]. These types are commonly recognised as the prevalent IVN attacks, frequently studied in experimental research, and often considered in the development of IDSs [33, 9, 34, 1, 35]. Consequently, this research focuses on these common and effective IVN attack types.

2.4.1 Injection Attacks

Injection attacks involve introducing new malicious frames into the CAN bus, thereby altering the inter-arrival time of frames or disrupting the sequential behaviour of CAN IDs. Common injection attacks on the CAN bus include:

- **DoS attacks:** DoS attacks try to make communication services unavailable by sending a large number of frames. In the context of the CAN bus, attackers can

Table 2.1: Experimental attacks on In-vehicle networks

Reference	Attack type	Attack surface	Violated security properties	Consequence
[33] 2015	Injection, Masquerade	Cellular network	Authentication, Confidentiality, Integrity, Availability	Engine Control, Break Control,Steering wheel control
[9] 2016	Injection	OBD-II	Authentication, Integrity, Availability	Steering wheel control, Engage brakes, Acceleration
[11] 2017	Injection	Wi-Fi, Cellular network	Authentication, Confidentiality, Integrity, Availability	Damage integrated circuit and gateways, Break control
[10] 2019	Injection	OBD-II, Cellular network, USB	Authentication, Confidentiality, Integrity, Availability	Gained control of the CAN bus

continuously transmit frames with low CAN IDs, particularly those assigned the highest priority. Koscher et al. [8] disabled the communication of individual components of the CAN bus using a DoS attack. Figure 2.7a illustrates a DoS attack on the CAN bus. The presence of the high-priority CAN ID 0x000 may introduce delays for frames with CAN ID 0x372, transmitted by ECU B. Such delays have the potential to induce unexpected behaviour in the vehicle.

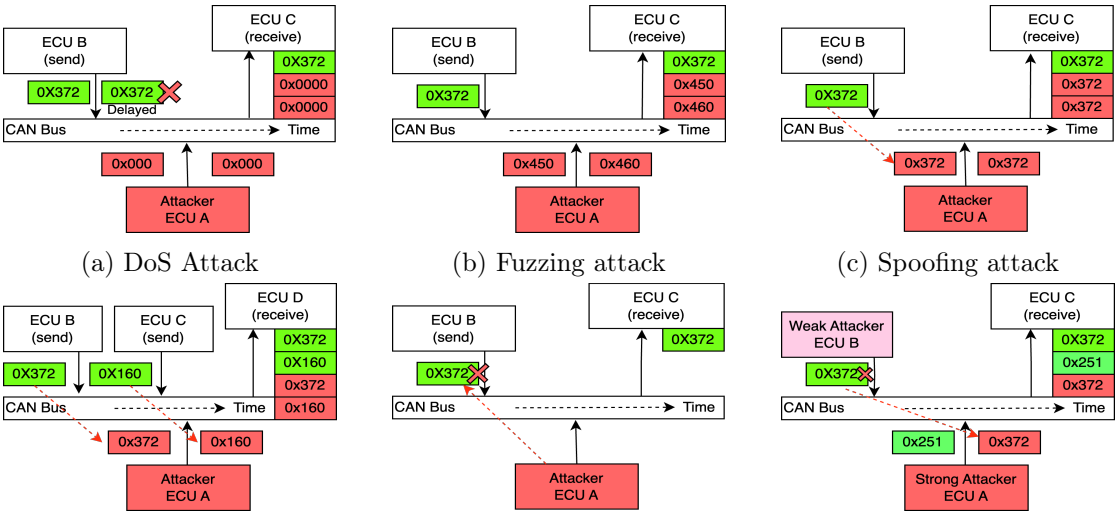
- **Fuzzing attacks:** In a fuzzing attack, a malicious node floods the network with a large number of frames, employing randomly generated IDs and malicious payloads to mimic legitimate frames. Two variations of this attack exist: injecting CAN IDs that appear during normal traffic (valid IDs) and injecting entirely new, randomly generated CAN IDs. The introduction of new CAN IDs was observed in the fuzzing attack conducted during a real CAN bus attack documented in [30]. The fuzzing attack on the CAN bus is depicted in Figure 2.7b. The attacker, ECU A, transmits randomly generated CAN IDs 0x450 and 0x460, causing the receiver ECU C to read and utilise information from these malicious frames. Attackers may execute this attack with prior knowledge of CAN frames, acquired through CAN bus sniffing or as a black-box attack without prior knowledge of CAN frames.

- **Spoofing attacks (Targeted ID attack):** In a spoofing attack, the attacker targets specific CAN IDs to introduce malicious messages. In [33, 36, 9, 37, 38], the authors used spoofing attacks in their experimental attacks on vehicle networks. Figure 2.7c illustrates the spoofing attack that attacker ECU A targets CAN ID 0x372 of ECU B. In this scenario, alongside the legitimate frame transmitted by ECU B, ECU C receives additional frames with the same ID but manipulated payloads. Consequently, ECU C might respond based on the malicious data.

- **Replay attacks:** In replay attacks, attackers capture and resend previously valid

frames at different times. For instance, previously recorded speedometer values may be transmitted at a later time. Kosher et al. [8] used replay attacks to control the radio and number of body control module functions in the CAN bus. Figure 2.7d illustrates a replay attack where attacker ECU A transmits CAN IDs belonging to both ECU B and ECU C.

Executing DoS and fuzzing attacks with randomly generated CAN IDs does not necessitate prior knowledge of the CAN bus. However, executing a fuzzing attack with existing CAN IDs and replay attacks requires limited prior knowledge, which can be acquired through CAN bus sniffing. In contrast, a spoofing attack demands advanced knowledge of the CAN specification, particularly when targeting specific vehicle functionalities.



(a) DoS Attack (b) Fuzzing attack (c) Spoofing attack

(d) Replay Attack (e) Suspension Attack (f) Masquerade attack

Figure 2.7: CAN bus attacks

2.4.2 Suspension Attack

In contrast to injection attacks, suspension attack do not introduce additional frames into the CAN bus. Instead, it involves compromising an ECU, preventing it from transmitting messages for a specific duration. To achieve this, the attacker, through a diagnostic session, manipulates the ECU into programming mode, rendering it unable to transmit messages. An illustration of a suspension attack scenario is presented in Figure 2.7e. In this scenario, the attacker compromises ECU B, suspending its transmission of frames with ID 0x372, thereby disrupting other ECUs reliant on messages from ECU B.

2.4.3 Masquerade Attack

In a masquerade attack, the attacker suspends an ECU and then utilises a strongly compromised ECU to transmit malicious frames with the same ID and frequency. For instance, as illustrated in Figure 2.7f, the attacker can monitor and learn about message IDs and their frequencies from the weak attacker ECU B (ID 0x372). Subsequently, the attacker suspends ECU B's message transmission, allowing ECU A to transmit a fabricated message representing ECU B. In [33], the authors employed a masquerade attack in their Jeep Cherokee attack, wherein they seized control of the steering wheel. As masquerade attacks neither inject new frames nor suspend frames with certain IDs in the CAN bus, they adhere to the frequency behaviour of the CAN ID.

2.5 Data Poisoning

Data poisoning represents a type of attack where adversaries manipulate a subset of the training data utilised to train AI models [39, 40]. This attack can manifest in two primary forms: dirty-label data poisoning and clean-label data poisoning [40]. Dirty-label poisoning entails altering both the labels and content of a fraction of the training data, whereas clean-label data poisoning solely modifies the content of the data, with the victim providing the labels. These poisoning strategies can further be classified as targeted attacks, which concentrate on specific datasets or classes, and untargeted attacks, which aim to increase the occurrence of erroneous predictions. Depending on the adversary's knowledge and objectives, these attacks can be categorized as white-box, black-box, or grey-box attacks. In a white-box attack, the adversary possesses complete knowledge of the target model or data, whereas in a black-box attack, this knowledge is absent. Grey-box attacks fall between white-box and black-box, with the adversary having partial knowledge about the victim model. Adversaries may employ various techniques, including data injection, data modification, label manipulation, and model tampering, depending on their capabilities. Label poisoning also known as label flipping, is a commonly used and realistic approach in data poisoning [40].

2.6 AI techniques for CAN Intrusion Detection

This section covers the AI techniques employed in this research to develop the AI-based IDS. Additionally, it discusses the evaluation metrics used to evaluate and compare the proposed models.

2.6.1 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) belong to a class of supervised machine learning models comprising artificial neurons with one or more feedback loops [41]. These feedback loops represent recurrent cycles over time or sequences. Training an RNN in a supervised manner necessitates a training dataset containing input-target pairs. The primary objective is to minimize the difference between the output and target pairs by adjusting the weights. However, RNNs encounter the vanishing gradient problem [42], wherein gradient magnitudes exponentially diminish as they propagate back through time [43]. Consequently, the network's memory tends to overlook long-term dependencies, struggling to discern correlations between temporally distant events.

Long Short-Term Memory (LSTM) models are tailored to mitigate the vanishing gradient problem by incorporating memory cells into their architecture. Unlike traditional sigmoid or tanh activations, these memory cells are governed by gates, which control the flow of information to hidden neurons, thereby retaining features extracted from prior time steps. A standard LSTM cell consists of input, forget, and output gates, in addition to a cell activation component. These units receive activation signals from various inputs and adjust cell activation using specific multipliers. The cell structure of a LSTM unit is illustrated in Figure 2.8. The LSTM formulation can be expressed by the following equations:

$$
\begin{aligned}
f_t &= \sigma\left(x_t W_f + h_{t-1} U_f + b_f\right) \\
i_t &= \sigma\left(x_t W_i + h_{t-1} U_i + b_i\right) \\
o_t &= \sigma\left(x_t W_o + h_{t-1} U_o + b_o\right) \\
\hat{c}_t &= \tan\left[x_t W_C + h_{t-1} U_c + b_c\right] \\
C_t &= \sigma\left(f_t \times C_{t-1} + i_t \times \hat{c}_t\right) \\
h_t &= \tanh\left(C_t\right) \times o_t
\end{aligned}
\tag{2.1}
$$

where x_t, h_t, C_t are the input vector, output vector and cell state vector, respectively. W, U, and b denote the weight matrices and bias vector parameters. Unlike traditional RNNs, LSTM gates possess the ability to prevent changes to the contents of memory cells across multiple time steps. This capability allows LSTM networks to maintain signals and propagate errors over extended sequences. Consequently, LSTM networks excel at handling data with intricate and distant interdependencies, rendering them adept at various sequence learning tasks. Therefore, LSTM can effectively learn CAN ID sequences.

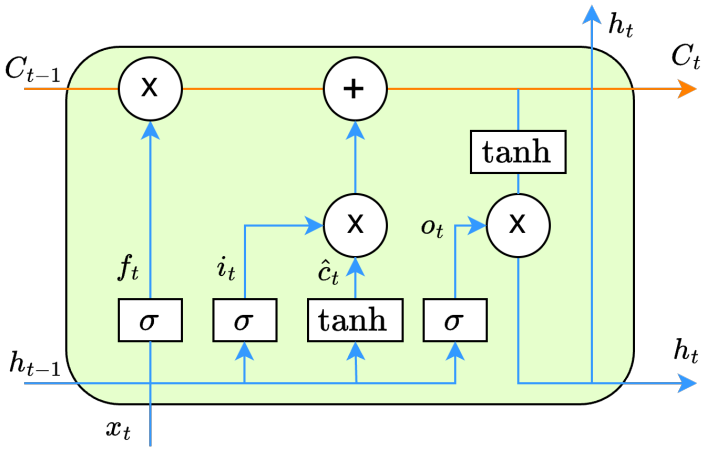


Figure 2.8: The cell structure of a LSTM unit

While LSTMs have proven effective in mitigating the vanishing gradient problem, they come with a higher memory requirement due to their multiple memory cells. An alternative solution is provided by the GRU, a variant of LSTM-based recurrent neural networks, which offers optimization. Like the LSTM, the GRU integrates gating units to regulate information flow within the unit, but it does so without requiring separate memory cells. Unlike the LSTM, the GRU exposes the entire state at each time step and computes a linear combination between the current state and the newly computed state. The cell structure of a GRU unit is depicted in Figure 2.9. The GRU formulation can be expressed by the following equations:

$$\begin{aligned} r_t &= \sigma\left(x_t W_r + h_{t-1} U_r + b_r\right) \\ z_t &= \sigma\left(x_t W_z + h_{t-1} U_z + b_z\right) \\ \hat{h}_t &= \tanh\left(r_t \times h_{t-1} U + x_t W + b\right) \\ h_t &= (1 - z_t) \times \hat{h}_t + z_t \times h_{t-1} \end{aligned} \tag{2.2}$$

where r_t, z_t, and $\hat{h}_t$ are the reset gate, update gate and candidate hidden layer, respectively. Given the GRU unit's lower memory demands compared to LSTM, the IDS introduced in Chapter 5 utilised GRU nodes to learn the CAN ID sequences.

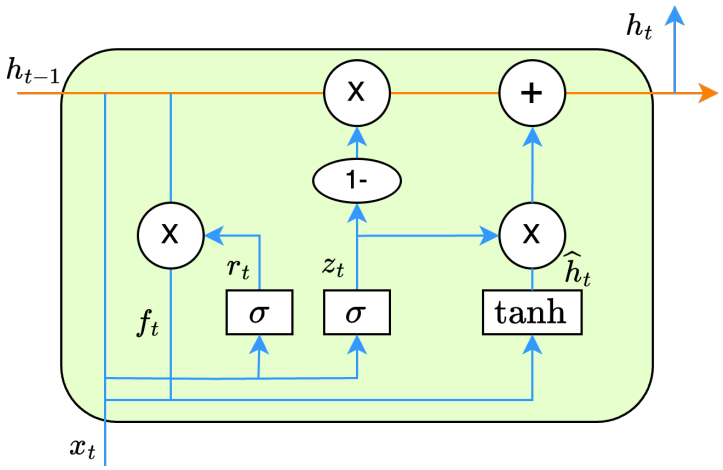


Figure 2.9: The cell structure of a GRU unit

2.6.2 Transfer Learning

Transfer learning is used to improve a learner for one domain by transferring information from a related domain. Typically, transfer learning is used when the training data is expensive or difficult to collect. Fine-tuning is the most commonly employed approach for transfer learning in deep learning models [44]. This technique allows for the enhancement of a pre-trained model within the same domain, either by introducing a new classification layer or by solely retraining the last layers of the model with additional data. The retrained layers, known as trainable layers, update their weight and bias parameters based on the new data, while the parameters of the frozen layers remain unchanged. This approach facilitates the retention of knowledge from the pre-training phase while enabling the model to adapt its parameters to better suit the new data. Restricting fine-tuning to only the last few layers aids in mitigating overfitting, a potential issue that may arise during full network retraining [44]. Thus, this method is employed in the CAN-ODTL model proposed in Chapter 6.

2.6.3 Autoencoders (AE)

AE is a feed-forward neural network which trains to reconstruct the input as the output. Generally, the vanilla AE consists of two parts, an encoder f_ϕ and decoder g_θ. The encoder maps the input space x to a lower dimensional hidden representation known as the latent space z. The decoder does the opposite by mapping the latent space to the output space $\hat{x}$ by approximating it to the original input space x. This procedure can be

formulated as follows:

$$z = f_\phi(x) \tag{2.3}$$

$$\hat{x} = g_\theta(f_\phi(x)) = g_\theta(z) \tag{2.4}$$

The objective of the AE is to train encoder f_ϕ and decoder g_θ to minimize the difference between input space x and output space $\hat{x}$ (reconstruction error). This is given by:

$$min_{\phi,\theta}||x - g_\theta(f_\phi(x))|| \tag{2.5}$$

where $||.||$ denotes the l_2-norm [45]. AE-based anomaly detection assumes that benign data have a smaller reconstruction error due to the learned patterns and anomalous data have a large reconstruction error. Therefore, in vanilla AE, reconstruction error is used as the anomaly score to distinguish benign and anomalous samples. The Latent AE, introduced in Chapter 7, employs AEs.

2.6.4 Mahalanobis Distance

Mahalanobis distance M_d is a distance measure which calculates the distance between a sample x and a multivariate distribution. It takes into account the correlation and variances of the variables, unlike the Euclidean distance which assumes uncorrelated variables with equal variances. This is given by:

$$M_d = \sqrt{(x - \hat{\mu})^T S^{-1} (x - \hat{\mu})} \tag{2.6}$$

where $\hat{\mu}$ and S^{-1} are the mean vector and inverse covariance matrix of the distribution. Mahalanobis distance is effective in detecting Out-Of-Distribution (OOD) samples [46]. It is particularly effective in identifying outliers within a multivariate distribution that involves correlated variables. It is therefore utilised to detect the data poisoning discussed in Section 6.5. By considering the underlying structure of the data, the Mahalanobis distance provides a robust and precise measure of the distance between a point and a distribution.

2.6.5 Attention Mechanism

Dealing with longer sequences or sentences poses a challenge for RNNs. To address this limitation, attention mechanisms were introduced, allowing access to all sequence elements at each time step. The transformer architecture, as described in [47], introduced the self-attention mechanism. Self-attention, also known as intra-attention, relates different positions of a single sequence in order to compute a representation of the sequence [47]. This enables the model to assign varying importance to different elements within an input sequence and dynamically adjust their impact on the output. Various self-attention variants have been proposed in the literature [48]. This thesis employs the original scaled-dot product attention mechanism because of its computational efficiency compared to other attention mechanisms like additive attention. It can be formulated as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.7}$$

where Q, K, V are the query, key and value matrices. d_k is the dimension of keys. A Self-attention layer is integrated into the CAN-ODTL model discussed in Chapter 6.3.

2.6.6 Model Quantization

Quantization is an optimization technique that reduces the precision of the numbers used for model parameters. Quantization leads to achieving an improved throughput and model compression by moving 32-bit floats into low-precision formats such as 16-bit floats or 8-bit integers (int8). Two primary approaches to quantization exist: quantization-aware training and post-training quantization [49]. Quantization-aware training applies quantization during model training to achieve a desired level of precision, allowing weights, activations, and gradients to be quantized to very low precision. In contrast, post-training quantization directly converts a pre-trained 32-bit floating point model to a lower bit-depth precision. However, quantization may marginally reduce model accuracy due to precision loss. This thesis employs post-training quantization for its simplicity, efficiency in terms of training time, memory overhead, and data consumption [50]. Both the CAN-ODTL and Latent AE models, introduced in Chapter 6 and Chapter 7, respectively, use model quantization.

2.6.7 Hierarchical Clustering

Hierarchical clustering is an unsupervised learning technique used to group similar objects into clusters. This can be categorized as agglomerative and divisive hierarchical clustering. Agglomerative clustering starts with each data point as a separate cluster and progressively merges similar clusters until a single cluster remains. Conversely, divisive clustering begins with all data points in one cluster, then splits clusters iteratively until each contains only one data point. Both methods yield a dendrogram showing the relationships between data points [51]. Various linkage-based algorithms like single, average, complete-linkage and Ward's method are employed to compute the similarity between clusters [52]. In Section 6.5, hierarchical clustering is employed to identify the ID clusters.

2.6.8 Evaluation Metrics

Evaluation metrics are used to evaluate the AI models. These metrics serve to measure the accuracy and effectiveness of prediction models through a variety of evaluation techniques.

Precision

Precision measures the proportion of accurate predictions made by a model, specifically capturing the instances where the model correctly identifies true positives (TP). It is calculated by dividing the number of true positives by the sum of true positives and false positives (FP), as depicted in Equation 2.8.

$$Precision = \frac{TP}{TP + FP} \tag{2.8}$$

Recall and True Positive Rate(TPR)

Recall evaluates the model's capacity to identify true positives for every category and assesses how well it performs in this regard. It is formally computed as the ratio of true positives to the sum of true positives and false negatives (FN), as illustrated in Equation 2.9.

$$Recall = TPR = \frac{TP}{TP + FN} \tag{2.9}$$

Accuracy

Accuracy represents the proportion of correct predictions made for the test data. It is formally determined as the ratio of true positives and true negatives (TN) to the total number of predictions, as demonstrated in Equation 2.10.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.10}$$

F1-score

The F1-score, derived from precision and recall, proves particularly valuable for evaluating the performance of models with unbalanced datasets. The formula for calculating the F1-score is depicted in Equation 2.11.

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.11}$$

True Negative Rate (TNR), False Positive Rate (FPR) and False Negative Rates (FNR)

TNR, FPR and FNR can be defined as:

$$TNR = \frac{TN}{TN + FP} \tag{2.12}$$

$$FPR = \frac{FP}{FP + TN} \tag{2.13}$$

$$FNR = \frac{FN}{FN + TP} \tag{2.14}$$

Mean Absolute Error (MAE)

MAE measures the difference between the predicted values and actual values. For a sample of n observations y $(y_i, i = 1, 2, ..., n)$ and n corresponding model predictions $\hat{y}$, the MAE is expressed as [53]:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.15}$$

Table 2.2: Key Technical Terms

Term	Definition
IVN	In-Vehicle Network, the networks that connect vehicle subsystems internally and to one another so that they can coordinate their functions.
ECU	Electronic Control Unit, a component responsible for controlling various functions within a vehicle.
CAN Bus	Controller Area Network, the primary protocol for IVN communication between various ECUs, known for its low cost, high speed, lightweight design, and simplified installation.
CAN Frame	A structured unit of communication in the CAN bus, comprising an identifier, control bits, data field, CRC (Cyclic Redundancy Check), and end-of-frame bits.
CAN ID	Identifier for CAN messages which also known as the arbitration ID.
CAN Payload	The data field within a CAN message that range from 0 to 8 bytes and may include sensor, category, constant, or cyclical counter data.
Injection Attack	A type of cyberattack where fabricated messages are injected into the CAN bus.
Suspension Attack	A type of cyberattack that stops messages with certain CAN IDs.
Masquerade Attack	A type of cyberattack where an attacker impersonates an authorized ECU by sending messages with a forged CAN ID.
IDS	Intrusion Detection System, a security mechanism designed to detect unauthorized or malicious activity within a network.
RNN	Recurrent Neural Network, an AI technique used for processing sequential data.
LSTM	Long Short-Term Memory, A type of RNN designed to effectively capture long-term dependencies by using memory cells and gates to regulate the flow of information, addressing the vanishing gradient problem common in traditional RNNs.
GRU	A type of RNN that uses gating units to manage the flow of information without the need for separate memory cells, offering a simpler and often more efficient alternative to LSTMs.
AE	Autoencoder, an AI technique used for anomaly detection by learning to compress and reconstruct data.
Model Quantization	The process of reducing the number of bits used to represent a model's parameters, making it more efficient for deployment on devices with limited computational resources.
Attention Mechanism	A mechanism that allows a model to weigh the importance of different elements in an input sequence, enabling it to focus on relevant parts of the sequence when generating predictions.
Data Poisoning	A type of attack where adversaries manipulate a subset of the training data utilised to train AI models.

2.6.9 Technical Glossary

Table 2.2, provides the technical terminology essential for understanding the development and implementation of IDS in In-Vehicle Networks (IVN). This glossary serves as a foundational reference, ensuring clarity and precision in the subsequent chapters.

2.7 Chapter Summary

The CAN bus is the primary protocol for in-vehicle networks (IVNs), enabling communication between various ECUs due to its low cost, high speed, lightweight design, and

simplified installation. ECUs performing time-critical functions are typically connected to the high-speed CAN bus, while those with less time-critical functions are connected to the low-speed CAN bus. During normal driving conditions, the transmission of CAN ID frames exhibits sequential behaviour, while sensor values in the payload field show time-series behaviour. However, the CAN bus has several security vulnerabilities, including a lack of authentication, the use of a broadcast domain, the absence of encryption, and ID-based priority. These vulnerabilities can be exploited by cyberattackers to compromise in-vehicle networks. Common attack types include injection, suspension, and masquerade attacks, which are frequently studied in IVN experimental research and are the focus of IDS development efforts. This chapter specifically examines these attacks and their characteristics to inform the development of effective detection methods.

Since this research focuses on AI-based IDSs, this chapter discusses the AI techniques used in subsequent chapters to develop IDS. Although various deep learning solutions can identify anomalies in sequential and time-series data, complex AI models are unsuitable for deployment on IVNs due to constraints like limited computing resources and the need for near real-time detection. Therefore, appropriate AI-based techniques such as RNN, transfer learning, AE, and model quantization are discussed, as these were employed in this research to develop a deployable AI-based IDS for detecting attacks on the CAN bus.

Chapter 3

Literature Review

Due to numerous vulnerabilities and potential cyberattacks, significant efforts have been directed towards protecting vehicles from security threats. Both detection and prevention mechanisms can be used to identify or prevent cyberattacks. However, detection strategies are more realistic in terms of the operational and economical realities [54]. As a reactive security measure, current literature predominantly concentrates on the development of IDSs for IVNs. This chapter conducts a literature review of IDSs designed to detect attacks on the CAN bus, specifically focusing on AI-based IDS. Additionally, it explores publicly available benchmark datasets used for evaluating these IDSs, highlighting their advantages and disadvantages, AI model resilience, and concluding with an overview of challenges and research directions. This chapter addresses the RQ1.

The main findings of this chapter were published in ACM Computing Surveys 2023 [17]

3.1 In-vehicle Network Cybersecurity

Addressing CAN bus vulnerabilities with any countermeasure necessitates consideration of the real-time data transmission requirements and the available limited resources to avoid overloading the bus. Solutions based on cryptography, intended to enhance security by ensuring confidentiality, integrity, and authentication (CIA), frequently demand additional computational resources in both the ECUs and the CAN bus controller. Alternatively, these solutions may introduce latency and increase the bus load [55, 1]. Alternatives such as redesigning the protocol by altering fields in the frame, segmenting the message into multiple frames, or introducing nodes and components to the bus for

additional capabilities are also costly to deploy and may lead to incompatibility with current vehicles. The use of Firewalls and Intrusion Prevention Systems (IPS) in external interfaces to block access to the bus is another option [1].

The primary limitations of these solutions lie in their high cost and potential incompatibility with existing vehicles. On the other hand, the development of IDS can be relatively inexpensive and can be deployed in current vehicles without incurring substantial additional costs.

3.2 Methodology for Literature Review

The papers reviewed in this chapter were selected using Preferred Reporting Items for Systematic reviews and Meta-Analyse (PRISMA) [56] protocol.

Eligibility criteria

- Papers published between 2016 and 2024 (Feb) were selected based on the scope of this thesis. Papers should make use of AI algorithms to detect attacks/anomalies in IVNs.
- Google scholar was used for the keyword search. The keywords used were: "in-vehicle intrusion detection machine learning", "in-vehicle attack detection machine learning", "in-vehicle intrusion detection", "in-vehicle machine learning attack", "in-vehicle cybersecurity survey", "controller area network IDS", "controller area network attack detection", "controller area network machine learning" and "in-vehicle network anomaly detection". These keywords were selected considering the focus of this work.
- Backward and forward snowballing [57] and recommendations given by Mendeley reference manager were also used to collect all the relevant references.
- Papers were included or excluded by reading the abstract and introduction considering the scope of this thesis. The final set of papers were selected so that each category listed in the taxonomy provided in Figure 3.1 had at least one and preferably a few representative papers.
- Selected papers were read thoroughly to evaluate their detection algorithms, features, used datasets, targeted attack types, performance, strengths, and limitations.

Risk of bias

Google scholar is considered a good starting point as it helps to avoid bias for any specific publisher [57]. This study selected google scholar as the search engine. Though this is a comprehensive review, there will still be good papers not selected as they are out of defined eligibility criteria. Only the papers written in the English language were considered. Due to these limitations, this work may have overlooked some important works.

3.3 CAN Intrusion Detection Systems (IDSs)

IDSs can be categorized into two categories as signature-based detection systems and anomaly-based detection systems based on the detection technique [17]. Signature-based detection relies on a predefined list of attack signatures, ensuring a low false-positive rate by accurately identifying previously known attacks. However, this approach falls short when confronted with novel or previously unseen attacks. Additionally, signature-based IDSs necessitate maintaining a potentially extensive database of known attacks on IVNs. On the contrary, anomaly-based IDSs model the normal behaviour of CAN bus data using known benign datasets. They leverage learned patterns or statistical metrics to identify anomalies, enabling the detection of novel attacks. However, anomaly-based techniques are susceptible to false positives. Further classification of anomaly detection-based IDSs includes statistical approaches, frequency or time-based methods, and AI-based approaches [1]. Notably, AI-based techniques have demonstrated success in identifying cyberattacks in automobiles, as evidenced by the work of various researchers [58, 59, 60, 26, 32].

The IDSs proposed in the literature for detecting attacks on the CAN bus can be classified according to the taxonomy depicted in Figure 3.1. The CAN bus is susceptible to various cyberattacks, including injection, suspension, and masquerade. Injection attacks alter the inter-arrival time of CAN frames and modify payload patterns based on malicious data. Suspension attacks also modify the inter-arrival time of a targeted ID and disrupt payload patterns. In contrast, masquerade attacks neither introduce new frames nor suspend existing ones on the CAN bus. Consequently, they do not induce changes in the inter-arrival time of the IDs. Identifying changes in the payload field is crucial for detecting masquerade attacks. However, if an attacker fails to synchronize the message frequency of the targeted ID accurately or uses additional frames to silence the target ECU prior to the attack, it is probable that alterations to the CAN ID sequences will become apparent.

The IDSs in the literature are designed to capture these changes in CAN bus traffic. Therefore, these properties were taken into consideration in the proposed taxonomy to classify existing works. These IDSs are developed based on features, including CAN ID (ID), CAN Payload (Payload), CAN frame, and Physical characteristics. CAN frame represents feature combinations of ID, Payload, DLC, and time. Physical characteristics encompass physical layer features such as voltage.

AI-based IDSs developed for the CAN bus leverage various AI algorithms, including traditional Machine Learning (ML) models, deep learning models, sequence learning models, and hybrid models. Traditional ML models, often referred to as shallow models, encompass algorithms such as Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Logistic Regression (LR), Naive Bayes (NB), and clustering, which have been studied for several decades [61]. An Artificial Neural Network (ANN) model associated with one or two hidden layers is considered a shallow learning method [62]. Deep learning-based models, known for their effectiveness in identifying complex patterns, have been increasingly employed in automotive cybersecurity research. These models include Deep Neural Network (DNN) [63], RNN like LSTM and GRU [64], Convolutional Neural Network (CNN) [65], Deep Belief Network (DBN) [63], AEs [66], and Generative Adversarial Nets (GAN) [67] to detect intrusions in IVNs. Sequence learning, a technique commonly used in Natural Language Processing (NLP) applications, finds application in IVNs where CAN data can be treated as sequential or multivariate time series data. Given that most CAN IDs are transmitted based on defined time intervals or as a sequence of events, this property is utilised to identify anomalies in such sequences. Recent literature has employed N-gram [58] and Hidden Markov Model (HMM)-based techniques [68] for identifying anomalies in the CAN bus. In the hybrid model category, which combines AI-based and rule-based (specification-based) approaches, there exist both strengths and limitations. Rule-based detection techniques tend to have a low false-positive rate and high efficiency [69]. On the other hand, AI-based techniques excel in identifying unknown attacks, even though they require more computing resources.

Both supervised and one-class learning approaches can be employed to train these algorithms based on the available training data. In supervised training, the algorithm learns from labelled data, whereas one-class algorithms grasp the behaviour, structure, and distribution of either benign or malicious data. In the context of CAN IDSs, one-class learning utilises only benign data to train the algorithm, given the greater accessibility to benign data compared to instances of attacks. A threshold is defined to detect anomalies. For example, the LSTM algorithm can be trained using only benign data without using

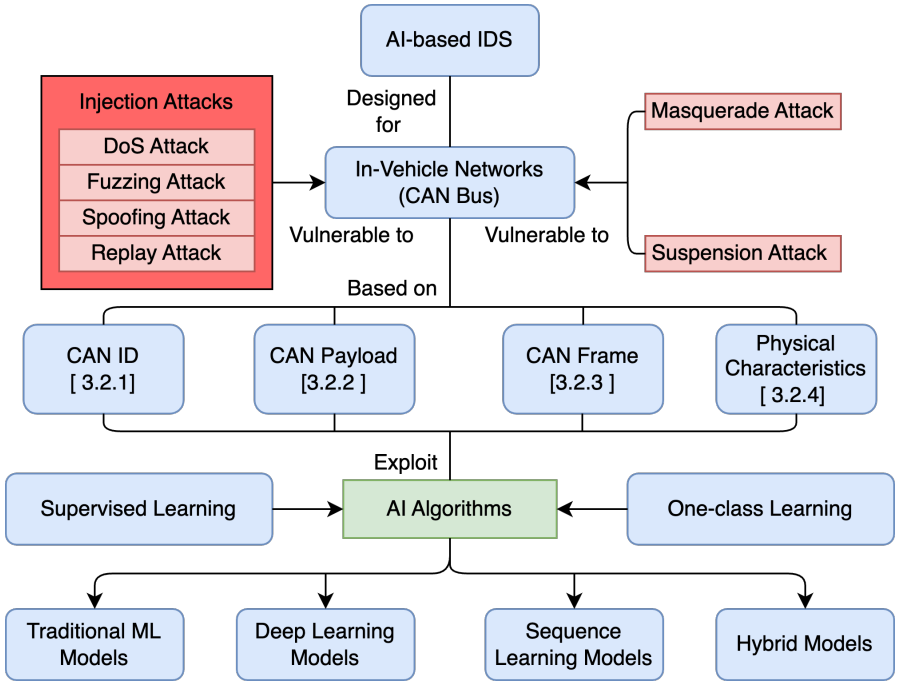


Figure 3.1: AI-based Intrusion detection system taxonomy for CAN bus. The numbers indicate the sections that cover each topic of the taxonomy.

labelled outputs [70]. Algorithms that exclusively use benign data during the training phase are categorized as one-class learning in the given taxonomy.

The subsequent subsections provide a literature review for each of the mentioned categories. Comprehensive summary tables for each subsection are presented in Table 3.1 to Table 3.7.

3.3.1 CAN ID-based IDS

Attacks such as injection and suspension alter certain properties of message ID sequences. These attacks can be executed by inserting or deleting frames, thereby changing the frame frequency compared to normal situations. Even in the case of a masquerade attack that does not alter the frequency of IDs, the sequence of the IDs may be modified due to a time synchronization mismatch with a legitimate ECU [71]. These distinctive properties can be utilised for the detection of attacks on the CAN bus. In the literature, authors have employed IDs as a feature in AI-based algorithms to develop IDSs. Features related to IDs were calculated using timestamps, time differences between consecutive IDs, and ID sequences.

One-class Learning based Methods

In [67], the authors introduced GIDS (GAN-based Intrusion Detection System) for IVNs, utilizing GAN to detect unknown attacks solely from benign data. GIDS employs two models: a generative model capturing data distribution and a discriminative model estimating sample probability within the training data. The system, evaluated on Hacking and Countermeasure Research Labs' car hacking dataset (HCRL CH) [72], achieved 100% average accuracy with the first discriminator and 98% with the second for DoS, fuzzy, RPM, and gear spoofing attacks. According to the authors, GIDS is difficult for attackers to manipulate due to the pre-trained deep learning method. A subsequent GAN-based IDS [73], replacing the GAN's true false classifier with double classifiers, outperformed the model introduced in [67] for all attacks. Another GAN-based model [74], focused on anomaly detection, leveraged CAN ID frequency as a feature, limiting detection capability for masquerade attacks.

The model proposed in [27] used a modified one-class support vector machine (OCSVM) with a bat algorithm for parameter optimization. Evaluation on CAN bus data, including public datasets [75], showed superior performance against DoS attacks compared to Isolation Forest (IF) and classical OCSVM benchmarks. Similarly, a frequency-based OCSVM model used in [76], employed social spider optimization for parameter optimization, yielding promising results against DoS attacks on real vehicle data. Despite their efficacy, both models need further testing against diverse real-life attack scenarios.

In [58], the authors proposed a context-aware anomaly detector for CAN bus cyberattack monitoring, utilizing N-gram distributions for sequence modelling. The model, based on maximum likelihood estimators and anomaly certainty ratios, demonstrated higher accuracy in identifying Revolutions per minute (RPM) and gear spoofing attacks using the HCRL CH dataset. However, the algorithm's computational efficiency and broader attack detection capabilities were not discussed. In [77], the authors employed a similar approach like [58] to develop a hybrid anomaly detection framework for diagnostics communication. However, their detection framework is only limited to automotive diagnostic communication. A temporal convolutional network-based IDS proposed in [78] utilised word embedding of CAN IDs, displaying effective detection of fuzzy and DoS attacks on the HCRL CH dataset. Treating CAN ID sequences as sentences, in [79], the authors introduced a Generative Pretrained Transformer (GPT) model focused on injection attacks, outperforming a unidirectional GPT model. Additionally, [80] applied a bag-of-words approach for intrusion detection, employing ML models with frequency

counts generated from sliding windows. In [59], the authors proposed an anomaly detection algorithm based on CAN ID recurring patterns. Their method involved creating a transition matrix representing all possible transitions between consecutive CAN IDs. In the attack detection phase, this validated transition matrix determined the status of consecutive ID sequences, classifying new IDs as normal or anomalies. Evaluation on a real dataset with various attacks revealed a low detection rate, particularly around 20%-40% for replay attacks, potentially resulting in a higher false-positive rate compared to [58], which classified messages based on a window rather than assigning labels for each message. The model proposed in [81] utilises Bidirectional Encoder Representations from Transformers (BERT) to identify anomalies in ID sequences. However, due to the complexity of BERT, this necessitates significant computing resources to make inferences.

In [82], the IDS employed an LSTM model assessed by comparing predicted IDs with actual IDs, achieving 60% accuracy. Their improved model [64] employed separate LSTM models for each ID, ensuring 100% detection for all attacks. In [83], the authors proposed an IF-based IDS for message injection attacks using CAN ID timing as features, demonstrating advantageous linear time complexity and low resource requirements, especially effective against gear and RPM spoofing attacks in a one-class training approach. In their work [84], the authors employed total counting and ID-counting features in a CAN sequence window for detecting malicious messages, utilizing both supervised and unsupervised classification methods.

Supervised Learning based Methods

Leveraging the sequential nature of CAN data, [85] introduced a deep convolutional neural network-based (DCNN) IDS to detect message injection attacks. Employing Inception-ResNet with a 29x29x1 input and binary output, the model outperformed baseline models across all attack types, demonstrating its effectiveness on the HCRL CH dataset. Similarly,[86] implemented a CNN-based IDS trained on recurrence plots, yielding a detection latency of 117ms on NVIDIA's Jetson TX2. Proposing a blockchain-based federated forest Software-Defined Networking (SDN)-enabled IDS, in [87], the authors developed an RF model for attack detection, utilizing Fourier transformation to generate features from CAN IDs. Evaluated on the HCRL OTIDS dataset [88], this model not only enhances AI model security by reducing the risk of adversary poisoning but also maintains data confidentiality for vehicle owners and manufacturers. In [89], the authors introduced a transfer learning-based self-learning IDS (TLSIDS). Comprising four

modules – the basic detection module, advanced detection module, unknown attacks classification module, and self-learning module – the TLSIDS aims to enhance intrusion detection capabilities. It adopted the same input data format as utilised in [85], employing a DenseNet-based detection model for identifying attacks.

In [90], a LSTM model detected malicious message injections in the CAN bus using message sequence graphs of CAN IDs. Features for the LSTM model were calculated based on Pearson and cosine similarities across successive time windows. Evaluation on a real vehicle dataset with fabricated RPM and speed messages revealed the algorithm's detection capability, influenced by the selected window size. A similar graph-based model is proposed in [91], using seven graph properties as features. SVM and KNN ML models trained on these features achieved over 95% F1-score for DoS, fuzzy, and spoofing attacks in the HCRL CH dataset.

A significant limitation of ID-based IDSs is their constrained capacity to identify attacks altering the message payload without affecting ID sequences or frequencies. Nonetheless, in the case of an attack solely manipulating the payload, alterations in CAN ID sequences may occur due to event-triggered messages within the CAN bus [92].

3.3.2 CAN Payload-based IDS

Given that the CAN payload carries the information for transmission, attackers in injection and masquerade attacks typically manipulate payload data, altering one or a few bits or bytes based on CAN data specifications. Consequently, the association between payload data or time series values will deviate from benign data. This section explores IDSs leveraging these properties to detect such attacks.

One-class Learning based Methods

In [95], LSTM and OCSVM were tested for anomaly detection in CAN frames using a real dataset and synthetic anomalies. OCSVM exhibited a 7% false-positive rate with a linear kernel, while the non-linear kernel incurred significant optimization time. LSTM outperformed OCSVM, yet the evaluation was limited to two attack types, lacking detailed breakdowns for each. In [96], the authors introduced a one-class compound classifier for attack detection in three distinct CAN IDs, primarily considering payload values. Fuzzing attacks were employed for testing, but the results yielded high false positives. Subsequently, in [97], the authors used an algorithm to concatenate the adjacent payload values of each ID based on their value changes. This reduces the dimensionality of

Table 3.1: Summary of ID-based attack detection in CAN bus using one-class based and supervised learning

Reference	**Model**	**Algorithm**	**Dataset**	**Attack**	**Strengths**	**Weaknesses**
[93] 2021	Traditional ML	DT, RF, XGBoost	Collected real data	Flooding, fuzzy, malfunction, replay	High detection rate, realistic attack scenarios	limited capability to detect CAN payload manipulation attacks such as masquerade attacks
[83] 2021	Traditional ML	Isolation Forest	Public real data (HCRL car hacking)	RPM and gear attacks	Used only benign data for training, linear time complexity, low memory requirement	Only tested for simple attack types
[26] 2022	Hybrid	GRU	Public real data (HCRL CH, HCRL SA, ROAD)	16 attacks including injection and masquerade attacks	High detection rate for a wide variety of attacks, near real-time detection	limited capability to detect attacks on high-frequent aperiodic IDs
[94] 2022	Deep learning	GAN, Autoencoder	Public real data (HCRL CH)	DoS, fuzzy, RPM and gear spoofing	Near real-time detection	limited to message injection attacks
[85] 2020	Deep learning (Supervised)	DCNN	Public real data (HCRL CH)	DoS, fuzzy, RPM and gear spoofing	Near real-time detection	Limited capability to detect CAN payload manipulation, limitation of detecting unknown attacks
[87] 2021	Traditional ML (Supervised)	RF (Using Federated learning)	public real data (HCRL OTIDS)	DoS, fuzzy, Impersonation attack	protecting the confidentiality of sensitive data	Communication cost and detection time
[90] 2021	Deep learning (Supervised)	LSTM	Collected real data	RPM and speed attacks	Near real-time detection	Only tested for simple attacks
[91] 2021	Traditional ML (Supervised)	SVM, KNN	Public real data (HCRL CH)	DoS, fuzzy, RPM spoofing	Feature extraction using benign data	Low detection rate for spoofing attack
[82] 2022	Deep learning (Supervised)	CNN	Public real data (HCRL CH)	DoS, fuzzy, RPM spoofing	High detection rate for injection attacks	High detection latency
[89] 2023	Deep learning (Supervised)	GAN	Public real data (HCRL CH)	DoS, Gear and RPM spoofing	Detection of unknown attacks	High detection latency

the payload of each ID. Pearson correlation was used to cluster the different fields and used the local outlier factor, compound classifier and OCSVM algorithms for the attack detection. However, the results were not acceptable to use in real-world situations due to the high false positive rate.

IDS proposed in [54] treated CAN bus messages as a time series ML problem. Vehicle movement was modelled as a sequence of states, with a sliding window assessing posterior probabilities. Anomalies were identified based on these probabilities and a threshold. Although this model performed well with limited anomalous states, identifying specific sensor data in CAN messages posed challenges. In [68], the authors introduced a HMM-based hybrid anomaly detection, using rule-based engines to monitor interfaces and generate events for HMM model training. Despite achieving high AUC and F1 measures, obtaining a complete list of events and attributes proved challenging.

In [98], the authors used a LSTM model to predict the next CAN payload for each ID. Log loss of each bit was considered to form the anomaly signal. A similar model was proposed by [99] to predict the CAN measurements such as RPM and break positions. Access to these measurements is challenging without having the DBC file. In [70], a separate LSTM model was used to predict the next payload of each ID and concatenated them using a joint latent vector. The authors used a real vehicle dataset with 13 IDs and a synthetic dataset (SynCAN) with 10 IDs for the performance evaluation. The used anomaly score was only feasible with a limited number of signal values. To train LSTM models, 5000 consecutive messages were considered. This will be computationally expensive for modern vehicles with a large number of ECUs. In [100], 81 payload signals were grouped into 32 subgroups based on the signal relationships and trained AE models for each group. However, it is important to note that this approach relied on pre-identified signals as input features, which are typically not accessible without obtaining the CAN DBC file. Similar GRU and LSTM AEs were used in [101, 102] to reconstruct the payload values of each ID. In [103], the authors improved the GRU-based IDS [101] by replacing the GRU with a LSTM and introducing a self-attention layer. In [104], the authors introduced an IDS using a temporal convolutional neural network. A decision tree-based classifier was used as the attack detector. All of these models [101, 103, 102, 104] trained separate models for each CAN ID. However, a major limitation common to all of these models was ignoring the interactions from other ID payload values. In particular, this might limit the detection of contextual anomalies. Moreover, these IDSs might require higher memory to store multiple IDSs trained for each ID. It was shown that the association of payload data of different IDs are useful in intrusion detection [102].

CANShield [66], an ensemble model employing multiple convolutional AEs, exclusively utilised high-priority signals to streamline model complexity. While exhibiting a notable detection rate for injection and masquerade attacks, this method's signal selection relies on semantic knowledge of CAN payload, limiting its generalization across diverse vehicles lacking corresponding CAN DBC files. Similarly, the AE-based IDS proposed in [105] leverages knowledge of CAN DBC files. This incorporation enhances explainability, facilitating the understanding of the signals or ECUs targeted in an attack. The LSTM-based IDS proposed in [106] adopted an approach of training distinct models for each ID, incorporating the present payload of the ID and payloads from other IDs within a specified time window. However, the model faces the challenge of potentially overlooking crucial associations within a narrow time frame by not including specific IDs. Conversely, selecting a broader window could escalate computational complexity and introduce variable noise from non-associated variables. Larger windows also necessitate extensive data for comprehensive learning of variabilities. In [107], an AE-based model and a Gaussian Mixture Model (GMM) were employed for intrusion detection. Diverging from the prevalent approach of using the reconstructed signal for anomaly detection, this model utilised the latent space as input to the GMM model. The evaluation utilised a real dataset from a Mercedes ML350 with DoS and fuzzy attacks. However, the dataset encompassed only four CAN IDs, potentially limiting the practical applicability of the model's results. An AE and attention mechanism-based IDS is introduced in [108]. This converts the hexadecimal payload into binary values and utilises a multi-layer denoising AE to obtain a hidden feature representation. Attention layers and fully connected layers are used to identify whether the message is abnormal or not. This IDS only considers the two CAN IDs from the HCRL OTIDS dataset for the proposed solution.

In [109], the authors introduced the Hybrid Similar Neighborhood Robust Factorization Machine Model (HSNRFM), enhancing feature representation by incorporating data fields of similar neighbours. The factorization machine model utilised second-order interaction features to predict the probability of anomalous outcomes, focusing on only two CAN IDs for training and evaluation. In [110], a density ratio estimation method leveraging a neural network (NN) was employed for change detection in packet frequency. Nevertheless, this model also restricted its evaluation to only three CAN IDs. The CNN-LSTM with attention mechanism-based IDS proposed by [111] used one-dimensional convolution to extract abstract features and bidirectional LSTM for capturing time dependence. Evaluations on the CAN Signal Extraction and Translation Dataset (HCRL-SET) with

simulated payload attacks showcased superior performance compared to baseline models. Notably, the model exhibited good detection times under attack conditions in a real vehicle, achieving detection within 5.7ms. However, the model had limitations, such as selecting a subset of signal values and disregarding payload correlations between different IDs.

Supervised Learning based Methods

In [63], the DNN-based IDS utilised CAN payload features, employing mode and value information for dimensionality reduction. Initial weights were set using a separate deep belief network (DBN), and a template-matching technique compared training samples and new CAN packets for attack identification. Simulation data with packet injection attacks demonstrated the DNN's superior performance over baseline models. In [28], a DNN and triplet loss network were proposed for real-time CAN bus anomaly detection, leveraging the distance between anchor, positive, and negative samples. However, both [63, 28] relied on mode and value information of CAN data and identifying these information are challenging without having the DBC file. In [112], a DNN-based IDS utilised Gradient Descent with Momentum and Adaptive Gain (GDM/AG) for improved efficiency and accuracy. They had access to the sensor values and used those as separate features. However, these values cannot be distinguished without having the DBC file or knowledge about the CAN payload.

In [113], the authors introduced the continuous field classification (CFC) algorithm to identify payload value alignments. A deep learning-based approach was then employed to detect anomalous fields. The evaluation, conducted using a dataset from a Renault Zoe electric car with manipulated signals, revealed that the CFC approach slightly outperformed the field classification obtained from the DBC file. However, these attacks are not realistic as they were created during the post-processing. This method does not capture interdependencies among variables. In [114], the authors used four k-nearest neighbour classifiers to detect various attacks on the CAN bus, showcasing distinct algorithms like fuzzy-rough k-nearest neighbours, discernibility classifier, and fuzzy unordered rule induction algorithm. Additionally, a decision tree model based on genetic programming (GP) was employed for intrusion detection in the CAN bus [115]. Feature and feature boundaries were determined using CAN DBC files, and the algorithm demonstrated comparable detection capabilities to ANN algorithms with significantly improved detection time across three datasets, including the HCRL CH dataset.

Table 3.2: Summary of Payload-based attack detection in CAN bus using one-class based learning (Part 1)

Reference	Model	Algorithm	Dataset	Attack	Strengths	Weaknesses
[95] 2016	Deep learning	OCSVM, LSTM	Collected real data (Michigan solar car data)	Fuzzing, misplaced packets	Used only benign data for training	High false positive rate, only tested for simple attacks
[54] 2016	Sequence learning	HMM	Collected real data (Honda, Toyota, Chevrolet)	Speed and RPM anomalies	Used only benign data for training	Only tested for limited anomalies, hard to identify specific sensor data of CAN message
[98] 2016	Deep learning	LSTM	Collected real data (19 hours drive of Subaru impreza high speed bus data)	Interleave, drop, discontinuity, unusual, reverse	Require no domain knowledge to train the algorithm	Treats each CAN ID's data sequence as independent
[116] 2018	Traditional ML	One class compound classifier	Collected real data (CAN log of few minutes drive)	Fuzzing	Used only benign data for training	High false-positive rate
[68] 2018	Sequential Learning	HMM	Simulation data	Out of order, out of context, USB firmware update, OTA malicious updates	Used only benign data for training, used adaptive threshold	Limited capability to generalize without having events and attributes
[70] 2020	Deep learning	LSTM	Collected real data and simulation data (Syn-CAN)	Plateau, continuous change, playback, suppress, flooding	Used only benign data for training, low false-positive and false-negative rates	Applicable for a limited number of signals
[99] 2020	Deep learning	LSTM	Collected real data	Packet injection	Used only benign data for training, near real-time detection	Limited generalization capability without CAN DBC file
[100] 2020	Deep learning	Autoencoder	Collected real data and simulation data (Syn-CAN)	Plateau, continuous change, playback	Used only benign data for training	Not suitable for deployment in resource-constrained environment, manual group selection
[101] 2020	Deep learning	Autoencoder	Simulation data (Syn-CAN)	Plateau, continuous change, playback, suppress, flooding	Used only benign data for training, simple model architecture	Inappropriate evaluation metric for imbalance dataset, ignore the signal dependencies
[103] 2021	Deep learning	Autoencoder	Simulation data (Syn-CAN)	Plateau, continuous change, playback, suppress, flooding	Used only benign data for training	Ignore the signal dependencies
[117] 2021	Deep learning	Autoencoder	Collected real data	Packet injection	Used only benign data for training	Ignore the signal dependencies, slow computation

Table 3.3: Summary of Payload-based attack detection in CAN bus using one-class based learning (Part 2)

Reference	Model	Algorithm	Dataset	Attack	Strengths	Weaknesses
[107] 2021	Deep learning	Autoencoder	Collected real data	DoS, Fuzzy	Used only benign data for training	Simplistic evaluation dataset
[97] 2021	Traditional ML	Load outlier factor, compound classifier, OCSVM	Data from two unmodified popular UK small family cars	Packet injection	Used only benign data for training	High false-positive rate
[106] 2021	Deep learning	LSTM	Public real data (HCRL CH)	Flood, replay, drop, spoof, fuzzy	Near real-time detection	Only considered continuous signal values, ignore ID correlations
[111] 2021	Deep learning	CNN-LSTM	Public real data (HCRL SET)	RPM and gear spoofing	Used only benign data for training	Low detection rate for gear attacks
[104] 2022	Deep learning	CNN	Simulation data (SynCAN)	Plateau, continuous change, playback, suppress	Used only benign data for training, time and memory efficient	Ignore the signal dependencies
[118] 2022	Deep learning	Autoencoder	Public real data (HCRL OTIDS)	Payload value change	Near real-time detection	Only tested for simple dataset and attack
[66] 2023	Deep learning	AE	Public real data (ROAD) and Simulation data (SynCAN)	Injection and masquerade	Higher attack detection	Depend on the knowledge of CAN DBC file
[105] 2023	Deep learning	AE	Collected real data	Fabrication, Masquerade, and Suspension	Use of explainable AI	Depend on the knowledge of CAN DBC file

Table 3.4: Summary of Payload-based attack detection in CAN bus using supervised learning

Reference	Model	Algorithm	Dataset	Attack	Strengths	Weaknesses
[63] 2016	Deep learning	DNN	Simulation data (200,00 packets using OCTANE simulator)	Packet injection	Low false positive rate, near real-time detection	Limited generalization capability without CAN DBC file
[114] 2017	Traditional ML	KNN	Public real data (HCRL CH)	DoS, gear and RPM spoofing, fuzzy	High detection rate for spoofing drive gear RPM gauge attacks	Low precision for fuzzy and DoS attacks
[112] 2019	Deep learning	DNN	Collected real data (300,000 packets)	Replay	High detection rate for replay attack	Limited generalization capability without CAN DBC file
[113] 2020	Deep learning	DNN	Collected real data	Payload value manipulation	High detection rate, explainability of the results	Only tested for simple simulated attacks
[115] 2022	Traditional ML	Decision tree, GP	Public real data (HCRL CH)	RPM and gear spoofing	Near real-time detection, memory efficient	Limited generalization capability without CAN DBC file

3.3.3 CAN Frame-based IDS

Beyond relying solely on individual features such as ID or payload, IDSs in the literature leverage a combination of features to encompass the evolving patterns in CAN data sequences. This has the advantage of detecting both ID changes and payload manipulation attacks. Other features combined with ID and payload are DLC and time (time gap).

One-class Learning based Methods

In [119], the authors assessed the performance of NN, LSTM, SVM, and OCSVM algorithms for attack detection. Results from the HCRL CH dataset indicated that NN outperformed the other models. To overcome computational limitations in IVNs, [120] proposed a mobile edge-assisted LSTM-based anomaly detection approach, achieving a real-time performance of 0.61ms with approximately 90% accuracy. In [121], the authors introduced an IDS integrating deep learning and sets of experience knowledge structures (SOEKS), demonstrating improved attack detection using real vehicle data. In [122], the authors proposed an unsupervised Kohonen Self-Organizing Map (SOM)-based anomaly detector for the CAN bus, showing superior performance compared to traditional approaches against various attacks. Additionally, [123] presented an ensemble hierarchical agglomerative clustering-based model for detecting malicious traffic in heavy-duty ground vehicles, exhibiting a higher detection rate with spoofed engine speed messages in an SAE J1939 protocol dataset.

In [124], a deep denoising AE-based model was proposed for the detection of injection attacks. To optimize the network structure, the authors employed an evolutionary-based optimization algorithm to mitigate premature convergence issues. Experimental results, utilizing the HCRL OTIDS and two real datasets, demonstrated the superior performance of the proposed model compared to selected baseline models. A hybrid approach incorporating a LightGBM-based supervised model and an AE-based unsupervised model was introduced in [125]. Features included time differences of consecutive CAN IDs, CAN ID values, and payload values. Experimental findings, leveraging the HCRL Survival Analysis (HCRL SA) dataset [126], indicated that the hybrid model outperformed the pre-trained LightGBM model. In [127], an LSTM-based anomaly detection algorithm was proposed to identify abnormal behaviour in the CAN bus. The model exhibited a high accuracy of over 90% in detecting anomalous data. However, its generalization to other vehicles proved challenging, with suboptimal performance observed in testing on additional vehicles. Enhancing feature processing in an LSTM model [128] achieved

effective detection of malicious activity using the HCRL CH dataset. Outperforming baseline models in terms of both detection rate and latency, the proposed model demonstrated its efficacy. Similarly utilizing the HCRL CH dataset, in [129], the authors presented an LSTM AE-based model. Employing packet count and bandwidth of outbound traffic within a fixed window as features, the model excelled in detecting injection attacks. In [130], an AE model was proposed with dedicated models for each CAN ID. Meanwhile, [131] introduced an improved Isolation Forest (IF) method with data mass for detecting tampering attacks. Evaluated in a simulation environment, the proposed method outperformed OCSVM and LOF algorithms. In [132], an IDS based on multiple observations HMM is introduced, utilizing benign CAN data. The system determines the anomalous state of a frame by assessing the probability of its occurrence at a given moment, considering factors such as the frame's timing, ID, and data domain. The performance of this approach was only evaluated against basic machine learning algorithms, serving as the baseline for comparison.

Supervised Learning based Methods

In [133], an IDS based on Gradient Boosting Decision Tree (GBDT) employed nine features for classification, including CAN message payload and entropy-based features. The experimental results demonstrated a true-positive rate of 97.67% and a false-positive rate of 1.2%. However, the evaluation focused on a basic attack scenario involving changes in CAN payload values. In [134], the authors introduced a context-aware IDS (CAID) framework using ANN, evaluating its performance in a real vehicle for chip tuning and power boxing manipulations. Although the model accurately recognized manipulated attacks, the experiment was done in a constrained environment, whereas the real-world environment might be quite different. Another ANN-based lightweight model proposed in [135] marginally outperformed baseline models. In [136], K-Nearest Neighbours (KNN) and Support Vector Machine (SVM) algorithms were proposed for clustering and classifying DoS and fuzzy attacks in the CAN bus. According to the experimental results, KNN outperformed SVM for both attack types in the HCRL CH dataset. However, the DoS detection rate was comparatively lower than the fuzzy attack. The IDS presented in [137] employs an ensemble of multiple ML models to detect CAN bus attacks. However, due to the utilization of six models in the ensemble, this approach is susceptible to false positives and is not conducive to near real-time attack detection. The IDSs discussed in studies like [138, 139, 140, 141, 142, 143, 144, 145, 137] represent fundamental comparisons of ML and DL models for CAN attacks. However, it's noteworthy that these

Table 3.5: Summary of CAN Frame-based attack detection in CAN bus using one-class based learning

Reference	Model	Algorithm	Dataset	Attack	Strengths	Weaknesses
[119] 2018	Traditional ML	OCSVM, SVM	Collected real data and public real data (Renault Zoe electric car data, HCRL CH)	DoS, fuzzy, RPM and gear spoofing	Used only benign data for training	Low accuracy for fuzzy, gear and RPM attacks
[119] 2018	Deep learning	LSTM	Collected real data and public real data (Renault Zoe electric car data, HCRL CH)	DoS, fuzzy, RPM and gear spoofing	Used only benign data for training	Low accuracy
[121] 2019	Deep learning	DNN	Collected real data	Replay, DoS, flooding	Can be applied to different vehicles	Only tested for simplified attacks
[122] 2020	Traditional ML	SOM and k-means	Collected real data	DoS,RPM and gear spoofing, fuzzing	High detection rate	Highly complex structure, Not measured computational cost
[124] 2020	Deep learning	Autoencoder	Collected real data and public real data (HCRL OTIDS)	Flooding, fuzzy and malfunction	Find the optimum network learning structure for a higher detection rate	Risk of finding a complex model structure
[127] 2021	Deep learning	LSTM	Collected real data	Random CAN payload values	Used only benign data for training	Limited generalization capability to other vehicles
[125] 2021	Hybrid	LightGBM and Autoencoder	Public real data (HCRL SA)	Flooding, fuzzy and malfunction	Used only benign data for autoencoder model training	Limited performance evaluation
[128] 2021	Deep learning	LSTM	Public real data (HCRL CH)	DoS, fuzzy, RPM and gear spoofing	Near real-time detection	Demands a large number of observations to obtain high detection accuracy
[132] 2023	Sequence learning	HMM	Simulation data	Replay and masquerade	High detection rate	Use of simplified attacks

models lack the ability to detect unknown attacks.

In [146], the authors proposed an IDS aiming to balance the efficiency of rule-based approaches with the high detection rates of DNN-based methods. The initial rule-based stage efficiently detects anomalies, with frames passing this stage forwarded to the DNN-based model for further identification. Evaluation against five attack types using three real datasets demonstrated high detection rates and low false-positive rates. However, specific evaluation results for the five attack types were not provided. Similarly, in [147], the authors introduced a hybrid approach for in-vehicle intrusion detection, specifically applicable to periodic messages, utilizing datasets from four real vehicles. In [148], a hybrid IDS capable of identifying both point and contextual anomalies was proposed. Eight classes of sensor data defined in [149] were employed, and a lightweight online detector of anomalies (LODA) served as the classification algorithm [150]. Synthetic CAN data with altered sequences were used for evaluation, revealing promising results for the simplified anomaly scenarios. In [151], the authors introduced a rule-based and random forest (RF) hybrid IDS using time interval, data field differences, and ID lag values as features. The RF model exhibited inferior detection capability compared to the rule-based approach.

In [152], the authors proposed an IDS based on LSTM, leveraging time-series features of CAN frames such as frame interval, ID, and payload values. In [153], a LSTM-based attack detection model was introduced, and evaluated with replay and amplitude-shift attacks on the HCRL CH and AEGIS repository datasets [154]. In [155], a novel RNN-based IDS optimized LSTM and GRU architectures using a simplified attention model to achieve lightweight design. The Random Forest (RF) algorithm was employed for classification. In [156], a GRU-based lightweight IDS was proposed, showing near real-time performance with a higher detection rate than baseline models. However, its reliance on supervised learning hampers its ability to detect novel attacks. In [157], an attention-based technique was used, employing attention and self-attention layers to capture important data parts and relationships. In [158], an IDS for the CAN bus was proposed based on LSTM. The model considered both binary and multi-class classification, using vanilla LSTM and stacked LSTM models to detect both point and contextual anomalies by incorporating CAN ID and payload information. In [159], the authors used a CNN model instead of the LSTM model proposed in [158]. However, due to the supervised learning approach, both models in [158, 159] lack the ability to detect unknown attacks. In [69], a CAN bus attack detection framework utilised rule-based and deep learning (LSTM) models, with the ensemble model achieving better accuracy than individual

models. The authors also introduced CANTransfer [160], a transfer learning-based IDS using convolutional LSTM (ConvLSTM), demonstrating effectiveness in detecting new attacks with one-shot transfer learning. Additionally, the deep transfer learning-based P-LeNet method in [161] outperformed baseline models, showcasing the advantages of transfer learning in reducing data collection needs for new attack types. In [65], the CAN ID field undergoes conversion into a color representation based on its hexadecimal value. Subsequently, an image is generated considering the timestamp and DLC. This approach is employed to train a lightweight CNN model for detecting DoS attacks. Despite utilizing a novel image generation technique for CAN data, the use of a supervised learning approach diminishes the model's generalization capability.

In [162], a privacy-preserving IDS is presented, employing a federated CNN model. This approach demonstrates resilience against non-independent but identically distributed clients and addresses the challenge of scarce training data. However, the evaluation of this system was conducted only using the HCRL CH dataset, with several ML models serving as baselines for comparison. The LSTM-based IDS proposed in [163] outperformed traditional ML models such as RF and XGBoost.

3.3.4 Physical Characteristic-based IDS

All the previous IDSs operated on data within the CAN data frame. In [166], a cloud-based cyber-physical IDS for vehicles was introduced, incorporating both cyber and physical features. Utilizing deep multilayer perceptron and LSTM algorithms, the system was tested on a robotic vehicle. Motivated by [167, 168], in [169], the authors proposed VehicleEIDS, an IDS relying on vehicle voltage signals. Leveraging unique voltage signals from ECUs, the authors extracted differential signals' time-domain features from two vehicles. Employing a deep Support Vector Domain Description (SVDD) model, VehicleEIDS demonstrated over 97% accuracy in distinguishing ECU voltage signals. Notably, it is the sole IDS capable of identifying the attack source and offers deployment advantages in existing CAN buses without protocol changes, avoiding additional bandwidth or computing resources. However, testing was limited to simple attacks like injection and replay.

Table 3.8 depicts the benefits and drawbacks of commonly used AI algorithms in in-vehicle IDSs.

Table 3.6: Summary of CAN Frame-based attack detection in CAN bus using supervised learning (Part 1)

Reference	Model	Algorithm	Dataset	Attack	Strengths	Weaknesses
[133] 2017	Traditional ML	Gradient boosting decision Tree	Collected real data (Alsvin CHANA car)	Change payload values	Low false-positive rate	Only tested for simple anomalies
[134] 2017	Traditional ML	ANN	Collected real data (2015 passenger vehicle)	Chip tuning, power boxing	Detect manipulated attacks with high accuracy	Results may not valid under real world environment
[136] 2018	Traditional ML	KNN, SVM	Public real data (HCRL CH)	DoS, Fuzzy	High detection rate for fuzzy attack	Low detection rate for DoS attack
[146] 2018	Hybrid	DNN and rule-based	3 Collected real data (Honda Accord, Asia Brand, USA brand)	Random attack, zero ID message, replay, spoofing, drop attack	Low false-positive rates for all datasets, near real-time detection	Limited evaluation results for attacks
[148] 2018	Hybrid	LODA and rule-based	Simulation data (CANoe)	Altered signals	Both point and contextual anomalies detection	Only tested for limited simplified anomalies
[152] 2018	Deep learning	LSTM	Collected real data	modified ID, data field and flooding attacks	Both ID and payload attacks detection	Only tested for limited simplified attacks
[155] 2019	Deep learning	LSTM, GRU, RF	Public real data (HCRL OTIDS)	DoS, fuzzy, impersonation	High detection rate, near real time detection	Limited types of attack detection
[144] 2020	Traditional ML	RF, bagging, ada boosting, NB, LR	Simulation data(16.5 million data records)	DoS, RPM, and gear spoofing, fuzzing	High accuracy for spoofing gear and spoofing rpm attacks	Low accuracy for fuzzy and DoS attacks
[141] 2020	Traditional ML	RT, RF, SGD, NB	Public real data (HCRL CH)	DoS, RPM and gear spoofing, fuzzy	100% accuracy for DoS,RPM, and gear spoofing attacks	Low accuracy for fuzzy attack
[153] 2020	Deep learning	LSTM	Public real data (HCRL OTIDS, AEGIS repository)	Replay, Amplitude-shift attack	High accuracy compared to other tested algorithms	Limited generalization capability without CAN DBC file

Table 3.7: Summary of CAN Frame-based attack detection in CAN bus using supervised learning (Part 2)

Reference	Model	Algorithm	Dataset	Attack	Strengths	Weaknesses
[158] 2020	Deep learning	LSTM	Collected real data (Toyota hybrid car data of 120 second drive)	DoS, fuzzing, spoofing	Both point and contextual anomalies detection	Only tested for limited attacks
[69] 2020	Hybrid	LSTM and rule-based	Collected real data	DoS, fuzzing, replay	Near real time detection	Only tested for 3 simple attack types
[160] 2020	Deep learning	LSTM and transfer learning	Collected real data	DoS, fuzzing, replay	Reduce the need for collecting a massive amount of data, Real time detection	Only tested for 3 simple attack types
[77] 2020	Sequence learning	N-Gram	Collected real data (BMW i3)	Replace, insert, exchange CAN messages	Both point and contextual anomalies detection	Limited to automotive diagnostic communication
[164] 2021	Deep learning	CNN and AGRU	HCRL OTIDS	DoS, fuzzy, impersonation	Achieved state-of-the-art performance	Lack of evaluation results for computational efficiency
[145] 2021	Traditional ML	DT, RF, SVM, MLP	HCRL OTIDS	DoS, fuzzy, impersonation	High detection rate for impersonation attack	Poor detection for fuzzy attack
[145] 2021	Deep learning	DBL	Public real data (HCRL CH)	DoS, RPM and gear spoofing, fuzzy	Provide more information about predictions	Higher epistemic uncertainty
[165] 2022	Deep learning	CNN and LSTM	Public real data (HCRL CH)	DoS, RPM and gear spoofing, fuzzy	High detection rate	Computational expensive model architecture
[156] 2022	Deep learning	GRU	Public real data (HCRL CH)	DoS, spoofing, fuzzy	Lightweight model	Limited types of attack detection
[147] 2023	Hybrid	DNN and rule-based	Collected real data (Honda accord, Honda civic, Ford fusion, Chevrolet volt)	Injection, drop, masquerade	Importance feature selection, high detection rate	Incapability to detect new attacks
[137] 2024	Traditional ML	KNN, DNN, RF, LGBM	Public real data (HCRL OTIDS)	Fuzzy, DoS, impersonation	high detection rate	Use of basic attack data and simple ML models

Table 3.8: Benefits and drawbacks of commonly used AI algorithms in in-vehicle IDSs

Algorithms	**Benefits**	**Drawbacks**
Traditional supervised ML	Achieves superior accuracy for known attacks, demonstrate computational efficiency, and performs well with small datasets.	Absence of the ability to detect unknown attacks that were not present in the training data.
OCSVM	Required only benign data to train the classifier.	Highly sensitive to hyperparameters nu and gamma, exhibits subpar performance with multivariate payload data when compared to deep learning models.
SVM	Performs well with small datasets.	Sensitive to kernel function parameters.
ANN	Capable of training with non-linear data.	Necessitates a dataset with a substantial number of samples typically in the range of tens of thousands to millions of data points depending on the problem complexity.
K-means	Class label not required (Unsupervised training).	Sensitive to outliers, Sensitive to parameter K.
LSTM, GRU	Feasible to train the classifier with only one-class (benign) data, effective for detecting anomalies in CAN ID and payload sequential data.	Long model training time, necessitates a dataset with a substantial number of samples, Higher detection latency.
DNN, CNN, BDN, GAN	Effective for detecting anomalies in multidimensional payload data.	Long model training time, necessitates a dataset with a substantial number of samples typically in the range of tens of thousands to millions of data points depending on the problem complexity, model complexity.
Autoencoder	Required only benign data to train the classifier, Capability to detect point, contextual and collective anomalies by learning variable associations, effective for payload data.	Computationally expensive, Necessitates a dataset with a substantial number of samples typically in the range of tens of thousands to millions of data points depending on the problem complexity, Over generalization issue for anomaly detection.
N-gram	Effective for detecting anomalies in CAN ID sequences, Context awareness, Required only benign data to train the classifier.	Inefficient for larger N.

3.4 AI Model Resilience

AI is rapidly revolutionizing the automotive industry, bringing sophistication and introducing new challenges. The incorporation of AI capabilities into modern vehicles not only enhances functionality but also introduces potential vulnerabilities and risks. AI models are susceptible to various adversarial attacks, which can be categorized into

training-phase and testing-phase attacks based on their types [170]. Training-phase attacks involve data poisoning, while testing-phase attacks include oracle and evasion attacks. In the event of an attacker compromising an IDS, it loses its ability to detect attacks. Consequently, it is crucial to prioritize the security of AI-based IDS during both the development and deployment stages. Despite this, only a few in-vehicle IDS proposed in the literature focused on the adversarial attacks on IDSs.

In [171], a CAN IDS utilised physical layer features of ECUs for attack detection, employing the Mahalanobis distance metric. Their study revealed the vulnerability of multi-frame-based fingerprinting techniques on the CAN bus to Hill-climbing style attacks. Such attacks allow adversaries to manipulate the number of attack frames, concealing the attacker ECU's identity and gradually manipulating the fingerprinting decision threshold. However, the proposed approach effectively addresses these adversarial attacks. In [172], two adversarial attack models, false data injection attack (FDIA) and fast gradient sign method (FGSM), were implemented, reducing the attack detection capability of an LSTM-based CAN IDS. Both methods modify the original training data. The LSTM model used is based on [173], achieving over 98% attack detection in time series CAN payload signals. FDIA and FGSM attacks slightly altered these payload signal values, rendering the LSTM-based detection model ineffective with accuracies of only 1.58% and 0.53% under FGSM and BIM attacks, respectively. To mitigate these attacks, a defense scheme was proposed, involving robust LSTM training that incorporates both poisoned and benign samples, validated against a separate validation set. The training continues until stopping criteria are met, resulting in a robust detection model impervious to FGSM and BIM attacks. This work underscores the limitations of deep learning-based detection models under adversarial training and emphasizes the importance of effective defenses against such attacks.

3.5 Benchmark Datasets

Data serves as the foundation of AI algorithms, with the accuracy of AI models heavily reliant on the availability and quality of data. This principle extends to AI-based Intrusion Detection Systems (IDSs) deployed in vehicles. This section delves into publicly accessible datasets suitable for training and evaluating in-vehicle IDSs.

Car hacking dataset for the intrusion detection (HCRL CH) [72]

This dataset, released by the Hacking and Countermeasure Research Lab for academic purposes, consists of 500-second benign data alongside four datasets representing distinct attack types: Denial of Service (DoS), fuzzing, and two spoofing attacks (RPM and gear). Each attack dataset comprises 300 instances of message injection lasting 3-5 seconds, captured over 30-40 minutes. Dataset attributes are: timestamp, CAN ID, DLC, payload, and label representing injected messages and normal messages. Dataset captured a fair amount of attack instances. These attacks significantly alter ID frequencies, rendering them easily detectable through frequency-based or sequence-based approaches. Experimental results from various studies consistently demonstrate high accuracy, achieving an F1-score of over 99% for all attacks due to the simplicity and unrealistic nature of the data [26, 58, 174]. While benign data collection occurred during driving, signal decoding revealed that the car was stationary during attack data collection [30]. Furthermore, benign data and attack data are stored in different file formats. These limitations render this dataset unsuitable for evaluating an IDS, especially those developed using AI techniques.

CAN dataset for intrusion detection (HCRL OTIDS) [88]

This dataset, developed by the HCRL in conjunction with their remote frame-based CAN IDS [75], employs one vehicle to gather benign, Denial of Service (DoS), fuzzy, and impersonation (masquerade) attack data. It stands out as the sole publicly accessible CAN dataset featuring remote frames and responses. The dataset comprises approximately 17 minutes of benign data and 18 minutes of attack data. Dataset attributes are: timestamp, CAN ID, DLC, and payload. However, unlike the car hacking dataset, it lacks labels (ground truth) as an attribute. Instead, the documentation provides attack injection intervals, though these are deemed inaccurate [30] and are insufficient for labelling fuzzy and impersonation attacks due to a lack of details such as injected IDs. Moreover, according to their documentation, the masquerade attack in this dataset does not match genuine masquerade attacks, as it involves message injection.

Survival analysis dataset for automobile IDS (HCRL SA) [126]

HCRL released this dataset with their frequency-based CAN IDS [175]. Notably, it stands as the only publicly available CAN dataset featuring real attacks on three vehicles. For each vehicle, the dataset encompasses benign data and three distinct attack types: flooding (DoS), fuzzing, and malfunction (spoofing) attacks. The total duration of benign

data is approximately 3 minutes, while the total duration of attack data is around 9 minutes. Attributes of this dataset are: timestamp, CAN ID, DLC, payload, and label representing injected and normal messages. However, it is essential to note that these attacks are basic and can be easily detected using frequency-based or sequence-based IDS due to their impact on significant frequency changes. Furthermore, the benign datasets related to each vehicle are limited to 60-90 seconds, which may not be sufficiently large for training a robust IDS.

Car hacking attack and defence challenge (HCRL CHDC) [176]

HCRL collected this dataset utilizing one vehicle for a competition focused on advancing attack and detection methodologies for CAN bus systems. The dataset comprises benign, flooding (DoS), spoofing, replay, and fuzzing attacks, with timestamp, ID, Data Length Code (DLC), payload, label, and sub-class (indicating attack type) as data attributes. Unlike other HCRL datasets where attack datasets were stored in separate files, here, both benign and four types of attacks coexist in the same file with 23 minutes of data. Despite the presence of benign data interspersed between attacks, the benign dataset is notably limited and may not offer sufficient data for effective algorithm training.

CAN signal extraction and translation dataset (HCRL SET) [177]

HCRL released this dataset to facilitate research in CAN analysis, particularly in signal extraction and translation. The dataset encompasses 56 CAN traffic logs obtained by periodically sending On-Board Diagnostics (OBD) queries during controlled driving sessions. It comprises 28 unique CAN IDs. Notably, this dataset does not include any attack data and information related to benign data.

SynCAN dataset [178]

This synthetic dataset was released with the CAN IDS CANet [70]. The primary objective of this dataset is to train unsupervised CAN IDS. Widely utilised in the literature for evaluating unsupervised payload-based IDSs [70, 100, 32], it stands out by providing signal values without the raw CAN data. This characteristic makes it particularly suitable for testing signal-based IDSs. The dataset comprises training data and six test datasets, featuring one normal dataset and five attack datasets. The attacks are categorized as plateau, continuous, playback, suppress, and flooding. Notably, these attacks are synthetic and cannot be validated for their impact on a real vehicle. It's worth mentioning

that this dataset encompasses only 10 CAN IDs with a maximum of four signals, which is relatively limited compared to modern vehicles.

TU Eindhoven CAN bus intrusion dataset [179]

This dataset, released by the Department of Mathematics and Computer Science at Eindhoven University of Technology, uses two cars and a CAN bus prototype to collect 19 minutes of benign driving data. The synthetic attack data spans 8 minutes and includes diagnostic, fuzzing, replay, suspension, and DoS attacks. However, the manipulation of CAN message timestamps during the post-processing stage makes this dataset unsuitable for testing CAN IDSs that rely on time as a critical feature.

CrySyS Lab dataset and CAN log infector [180]

This benign dataset published by the Department of Networked Systems and Services at Budapest University of Technology and Economics represents various driving scenarios, including constant speed driving, lane changes, and emergency braking. The authors provided a CAN log infector that can be used to simulate a wide variety of masquerade attacks. However, introducing attacks during post-processing diminishes the behaviour of realistic attacks.

AEGIS Big data project [154]

Released as part of the "AEGIS-Advanced big data value chain for public safety and personal security" big data project, this dataset consists of benign data covering 20 hours of driving. It includes signal data such as wheel speed, steering wheel angle, roll, pitch, accelerometer values per direction, and GPS data. Similar to the SynCAN dataset, this one also provides signal values. However, the absence of attack data restricts the usage of the dataset for IDS evaluation.

Real ORNL Automotive Dynamometer (ROAD) CAN intrusion dataset [30]

This real dataset includes an advanced set of attacks, comprising 13 unique attacks and 12 benign datasets covering various driving scenarios. Data collection involved a single vehicle and included fuzzing, targeted ID (fabrication), and accelerator attacks. Fuzzing attacks introduced random IDs, while targeted ID attacks incorporated four variations: correlated signal, max speedometer, max engine coolant temperature, and reverse light.

Accelerator attacks induced a compromised mode in the ECU. Masquerade attack versions were generated for each targeted ID attack by filtering out legitimate messages during post-processing. While labels are absent, attack IDs and intervals are provided, aiding in identifying attack messages. Regarded as one of the most comprehensive CAN datasets, it enables the evaluation and comparison of CAN IDSs against realistic attacks. However, despite its advantages, this dataset has several drawbacks. Benign data collection involved both roads and a dynamometer, whereas during attack data collection, the vehicle was exclusively on a dynamometer. This variance in data collection environments may introduce discrepancies compared to actual road driving scenarios. Additionally, intentional changes were made to the order of CAN IDs during the obfuscation process, resulting in the removal of priority information. This limitation restricts the applicability of the dataset for IDSs reliant on ID priority information. Furthermore, although the dataset comprises 106 CAN IDs in the vehicle, during targeted ID attacks, only two high-priority IDs and one low-priority ID were targeted. This limitation impedes the evaluation of IDS capability to detect attacks on various IDs, particularly those of low and medium frequency. Considering the 106 IDs, acquiring a large dataset is essential to effectively learn the normal behaviour of the vehicle, surpassing the available 3-hour benign dataset. Additionally, attack datasets last only a few seconds for each targeted ID attack, imposing constraints on the thorough evaluation of an IDS.

3.6 Research Gaps and Challenges

Despite the growing focus on and publications of IDSs for the CAN bus, the progress in IDS research encounters significant challenges and limitations. This section identifies these challenges and limitations of current approaches.

Availability of benchmark datasets

The performance of an AI-based algorithm heavily relies on the quality of the data it uses for the model training, as low-quality data can yield suboptimal results. The advancement of IDS research faces significant obstacles due to the lack of high-quality, publicly available real CAN data that includes realistic attack scenarios [30]. Generating real attack data on moving vehicles involves substantial costs and risks. Utilizing a real CAN attack dataset for model development, validation, and testing is pivotal for developing an effective IDS capable of detecting diverse attacks in real-world scenarios. However, numerous proposed IDSs rely on proprietary datasets inaccessible to other researchers [181].

Publicly accessible CAN bus attack datasets, as discussed in Section 3.5, exhibit limitations such as insufficient data for effective learning of normal behaviour, a focus on only a few CAN IDs during attacks, significant variations in driving conditions between benign data collection and attack data collection, and the use of high-frequency injection for attacks, rendering them easily detectable even with simple time-based detectors. Notably, none of the existing attack datasets targets moving vehicles in realistic driving scenarios. Consequently, there is a clear need to generate a realistic attack dataset using a moving vehicle to facilitate comprehensive testing of various techniques and thereby enhance the comparison and validation of CAN IDS. This thesis addresses this research gap by introducing a CAN bus attack dataset in Chapter 8.

Model Generalisation

Due to the confidential and proprietary nature of CAN bus data specifications, which vary depending on the vehicle make, model, and year [182], developing IDSs with high generalization capability, especially for payload-based IDSs, poses significant challenges. However, IDSs that utilise the CAN ID field can learn relevant patterns without knowledge of CAN data specifications. As a result, CAN ID-based IDSs can be effective in detecting attacks like injection and suspension attacks, which typically alter ID patterns. Payload-based IDSs leveraging DBC file details have demonstrated higher detection rates by focusing solely on essential signals [66], facilitating the creation of lightweight solutions. Nonetheless, these IDSs lack generalizability across different vehicles without their respective DBC files. Future research could explore developing payload-based IDSs assuming no prior knowledge of CAN specifications or creating CAN field classification algorithms to accurately identify payload variables for IDS development. In Chapter 5, CAN ID-based IDS is introduced to address RQ2, enhancing the model generalisation capability. Additionally, Chapter 7 proposes CAN payload-based IDS to address RQ3, improving the model generalisation by leveraging the associations among byte-level variables in the payload field.

Detection latency

Message transmission within IVNs occurs in real-time, with approximately 0.5ms between two consecutive messages, necessitating IDSs capable of promptly detecting and implementing countermeasures. However, most examined deep learning-based literature struggled to achieve real or near real-time attack detection. While deep learning-based IDSs can leverage high computational resources in the cloud to enhance detection times,

the dynamic nature of vehicles introduces connectivity stability challenges in cloud deployments. An alternative worth exploring is edge computing, despite the inherent computational constraints. Future research could delve into conducting diverse experiments under real-world conditions to address these challenges and optimize the efficiency of IDSs for IVNs. Chapter 9 focuses on model deployment, aiming to answer RQ3 and bridge this gap by integrating lightweight IDSs into an edge device to achieve near real-time attack detection.

Evaluation metrics

In the literature, IDSs assessed their proposed models using various data sources, including collected real data, public real data, or synthetic data. The challenge arises from the different adversarial settings under which the evaluation of collected real and synthetic data occurred, making a uniform security comparison challenging. While performance comparisons for benchmark real or synthetic datasets are feasible due to a shared dataset, the lack of common evaluation metrics complicates the assessment. Some works used accuracy, precision, F1-score, or recall individually, making a comprehensive comparison difficult. Visual evaluations, such as bar or line charts, were presented in some works without providing the accurate comparable numerical figures, adding complexity to model comparisons. For a fair assessment, it is crucial to employ multiple metrics such as F1-score, precision, recall, TNR, FPR and FNR. Given the imbalanced nature of discussed attack datasets, accuracy is not an inappropriate metric. Additionally, considering detection latency is essential for in-vehicle IDSs, yet only limited works have evaluated models in this regard, accompanied by a discussion of the experimental platform. Incorporating these metrics into the evaluation criteria facilitates the identification of more effective methods and the enhancement of attack detection in IVNs. This thesis employs multiple metrics to evaluate the model performance, enabling accurate comparison of results in future studies.

One-class learning

One-class learning is particularly well-suited for the CAN bus, given its predictable and consistent data flow [96]. Additionally, due to the higher cost associated with collecting attack data in vehicle networks compared to benign data, one-class learning offers an efficient approach. In this paradigm, only benign data is utilised to model normal behaviour, and a threshold is established for anomaly detection. However, a significant

limitation of this approach is the requirement for a large dataset that adequately represents the normal profile to minimize false positives. A potential future research direction to address this limitation is streaming learning, allowing continuous adaptation of the model to evolving normal behaviour without excessive computational resources. Deploying the model in a vehicle and updating parameters and thresholds over an extended period can encompass diverse normal driving conditions effectively. The IDSs developed to address RQ3 and RQ4 in Chapter 5 and Chapter 7 solely utilise benign data for model training. The on-device transfer learning approach introduced in Chapter 6 addresses the challenges associated with one-class learning.

Requirement of large datasets

AI algorithms typically demand a substantial dataset for effective model training. However, as previously highlighted, the scarcity of realistic attack and benign datasets poses a significant limitation in the field of in-vehicle network attack detection. Confronting the challenge of learning from a limited number of examples is crucial. Approaches such as transfer learning [183], one-shot learning [184], and zero-shot learning [185], which have been successfully employed in domains like image recognition and NLP applications, offer potential avenues for future research. Exploring their adaptation to vehicle network data could prove instrumental in utilizing small datasets for the detection of new types of attacks. Chapter 6 addresses this gap by introducing a streaming learning approach, while the Latent AE model proposed in Chapter 7 relatively minimizes the need for large datasets by focusing only on important variables and removing unassociated ones. Additionally, Chapter 8 introduces a large benign dataset for one-class model training.

Cost of model deployment

Given the high cost and potential incompatibility with existing vehicles, IDSs present a more flexible option for IVNs compared to cryptographic-based solutions. Nevertheless, the majority of the reviewed literature on IDS development has not sufficiently addressed deployment requirements and countermeasures. In vehicle networks, ECUs contend with limitations such as restricted memory storage, computing power, and the CAN bus having a limited bandwidth. The development and deployment of IDSs are constrained by these resource limitations. IDSs can be deployed as host-based IDSs or network-based IDSs. Host-based IDSs prove impractical for vehicles, as they necessitate ECU modifications that are not cost-effective. Therefore, deploying a network-based IDS as an additional node in the CAN bus emerges as the most appropriate solution. Alternatively, considering

cloud deployment for IDSs presents another viable option. Chapter 9 addresses RQ4 by bridging this gap through the integration of IDSs into low-cost Raspberry Pi devices, thereby deploying as network-based IDS.

Protecting IDS

While AI-based models excel at detecting anomalies in vehicle networks with a high success rate, they are susceptible to cyberattacks, including data poisoning, oracle, and evasion attacks. Notably, the discussed literature lacks a focus on safeguarding their proposed models from cyberthreats, with the exception of models introduced by [87] and [186]. In [87], the authors suggested leveraging blockchain technology to enhance IDS security, whereas in [186], the authors proposed a defense mechanism against adversarial attacks on LSTM-based IDS. Despite the suitability of streaming learning for training one-class based CAN IDS, they remain highly vulnerable to data poisoning attacks [187]. Developing a secure IDS for IVNs in an adversarial setting represents a challenging and crucial avenue for future research, with potential adaptations of solutions employed in other domains. This thesis bridges this gap by introducing a data poisoning attack detection technique in Section 6.5. However, model tampering attacks remain unaddressed and are considered as future work.

3.7 Chapter Summary

In this chapter, a comprehensive review of current research and future research directions in in-vehicle network security, with a specific focus on CAN bus cybersecurity, has been presented. Considering the high cost and potential incompatibility with existing vehicles, IDSs emerge as a more adaptable option for IVNs compared to cryptographic solutions. Among these, AI-based IDSs have made promising strides in detecting various attacks on the CAN bus.

The IDSs discussed in the literature leverage features such as CAN ID, payload, CAN frame, or physical characteristics to train AI models. These models can be categorized into supervised and one-class learning, with the latter exhibiting superior capability in detecting unknown attacks. One-class learning algorithms, requiring only benign data for training and threshold estimation, offer a promising approach as benign data collection is more accessible than acquiring attack data in vehicle networks. AEs, particularly in deep learning, are commonly employed as effective one-class learning approaches for CAN payload-based IDS. Variants of Recurrent Neural Networks (RNNs), such as LSTM and

GRU, have proven effective for CAN ID-based IDS, utilizing only benign data. Combining LSTM or GRU with other deep learning algorithms, such as CNNs, AEs, or rule-based models, has enhanced attack detection capabilities across a wide range of scenarios. However, one-class-based deep learning models necessitate a substantial amount of data for model training to learn the normal behaviour of IVNs. This tendency may result in higher false positives compared to supervised learning. To address this, streaming learning with a large dataset and a window-based detection approach can be employed to reduce false positives. While deep learning models generally achieve better accuracy than traditional ML models, concerns regarding high resource requirements and detection latency persist, given the limited resources available in in-vehicle network devices. Hybrid and ensemble models have augmented detection power, leveraging performance improvement while mitigating individual model weaknesses. Some works have explored innovative approaches such as transfer learning, GANs, and federated learning, yielding promising results in terms of accuracy, new attack detection, and model security. Given the distinct characteristics of different attacks and deployment environments, an IDS employing multiple methods is essential to cover a broad spectrum of attacks. In light of the reviewed literature, one-class learning-based models, specifically focusing on different fields of the CAN frame, such as ID and payload, emerge as ideal algorithms for detecting a wide variety of attacks, considering the limitations of IVNs.

Despite the increasing attention and publications on IDSs for the CAN bus, progress in IDS research faces significant challenges and limitations. These include the need for benchmark datasets featuring realistic attacks to effectively evaluate developed models, enhancements in low-frequency attack detection, improvements in detection latency to meet the near-real-time requirements of IVNs, the utilization of thorough evaluation metrics for more effective model evaluation and comparison, addressing the substantial dataset requirements of one-class learning, reducing the cost of model deployment, and ensuring the protection of these IDSs against adversarial attacks such as data poisoning, oracle, and evasion attacks.

These findings inspire and influence our contributions and methods in the following chapters.

Chapter 4

Research Methodology

This chapter discusses the overall research methodology of this study, encompassing research design, model development, threat model and dataset, validation and evaluation, ethical considerations, assumptions, and limitations.

4.1 Research Design

To proactively address the growing concern of cyberattacks on vehicle networks, the current body of literature primarily emphasizes the development of IDSs [1]. The research design for developing an IDS for the CAN bus involves several key phases. First, a comprehensive literature review is conducted following the PRISMA protocol to analyse existing IDS solutions for CAN networks, identifying gaps and limitations in current approaches (Chapter 3). Next, threat modelling is performed to identify and categorise potential attacks on the CAN bus, such as injection, suspension, and masquerade. This process develops a detailed threat model to guide the IDS design (Chapter 2). The IDS architecture will then be proposed, consisting of modules for data collection, preprocessing, feature extraction, detection, and alert generation, and various AI algorithms (e.g., LSTM, GRU, AE) will be experimented with to develop the detection models (Chapter 5,6,7). Dataset collection will involve using publicly available benchmark datsets to study CAN traffic, including normal and attack scenarios, and collaborating with MIRA to collect real-world CAN bus data from vehicles under normal and attack conditions (Chapter 8). Feature extraction will analyse the collected CAN traffic to identify relevant features that distinguish normal behaviour from malicious activities, such as message ID frequency, inter-arrival times, and payload analysis. AI algorithms will be

trained using a portion of the collected dataset to recognize patterns associated with normal and malicious CAN traffic, and performance evaluation will validate the models using a separate dataset to ensure robustness and generalization. The IDS will be evaluated using metrics such as detection rate, false positive rate, false negative rate, computational overhead, and response time under different network loads and attack intensities (Chapter 5,6,7,9). The expected outcomes include a comprehensive understanding of CAN vulnerabilities and potential attack vectors, a novel IDS architecture specifically designed for CAN bus, AI algorithms capable of accurately detecting various attacks on the CAN bus, and performance metrics demonstrating the efficacy and efficiency of the proposed IDS (Chapter 9).

4.2 Model Development

Based on the literature review in Chapter 3, the sequential behaviour of CAN IDs and the time series behaviour of the payload field can be utilised to detect attacks on the CAN bus. As illustrated in Figure 4.1, this thesis first develops CAN-CID, a GRU-based model designed to detect anomalies in CAN ID sequences. The details of this CAN ID-based IDS are discussed in Chapter 5. The GRU model is chosen for its computational efficiency compared to RNN models like LSTM, enabling near real-time detection. This model requires a large benign dataset for training. To address this challenge, CAN-ODTL, an on-device transfer learning technique, is introduced in Chapter 6. This technique incrementally retrains the algorithm with streaming CAN data. CAN-ODTL retrains only the last layer of CAN-CID model to optimize retraining time and prevent overfitting. Since streaming learning models like CAN-ODTL are susceptible to data poisoning attacks, a data poisoning defense procedure is introduced in Section 6.5. This procedure employs the Mahalanobis distance for anomaly detection due to its superior capability in handling multivariate data. To detect sophisticated masquerade attacks that do not alter CAN ID sequences, a CAN payload-based model is necessary. Accordingly, Chapter 7 introduces Latent AE, a lightweight AE-based IDS that utilizes multivariate payload data to detect attacks. The Raspberry Pi 4 Model B is selected as the edge device for deploying CAN-ODTL and Latent AE models. This choice is based on the device's low cost and sufficient computational capability, enabling the deployment of these lightweight models to achieve near real-time detection latency.

CAN-CID, CAN-ODTL, and Latent AE are developed to achieve high attack detection rates with near real-time detection latency. Therefore, lightweight model architectures

are chosen to facilitate deployment on resource-constrained edge devices. Quantization techniques are applied to the CAN-ODTL and Latent AE models to enhance detection latency and reduce model sizes for edge device deployment. Grid search is employed to optimize parameters such as the number of layers and nodes in each layer of the proposed models. Given the limited availability of attack instances and the need to improve generalization capability, only benign datasets are used for training the proposed models. Data poisoning attack detection, as discussed in Chapter 6, utilizes synthetic poisoned data designed to mimic realistic injection attack characteristics. During model training, parameters such as batch size, optimizers, learning rate, number of epochs, and early stopping criteria are selected through repeated experiments to achieve optimal results.

Since the proposed models are based on anomaly detection techniques, different thresholds are required for effective attack detection. Separate benign datasets are used to calculate these thresholds, aiming to maximize attack detection while minimizing false positives. The proposed models are evaluated using a wide variety of attacks from benchmark datasets and realistic attacks generated during the creation of the CAN bus attack dataset, as discussed in Chapter 8. Metrics such as F1-score, TP, TN, FP, and FN rates, along with visualizations like bar and line charts, are used to interpret the results and make unbiased evaluations. All models are compared with appropriate baseline models to assess their effectiveness in terms of detection accuracy and latency. Python and TensorFlow are employed for model development and experimentation. For model training and accuracy evaluations, a MacBook laptop and Google Colab with GPUs are used, while a Raspberry Pi 4 Model B is utilized for CAN-ODTL, Latent AE, and model deployment detection latency evaluations. TensorFlow Lite converter is used to convert the model into a compressed flat buffer.

4.3 Threat Model and Datasets

As discussed in the Section 3.5, several publicly available CAN bus datasets are available to evaluate the CAN bus IDSs. The ROAD dataset, collected from a passenger vehicle, includes a variety of physically verified CAN attacks. Due to its advanced set of realistic attacks, including injection and masquerade, it is used as the primary dataset to evaluate the proposed CAN-CID, CAN-ODTL, Latent AE, and data poisoning experiments. This dataset was collected using SocketCAN software on a Linux computer with a Kvaser Leaf Light V2 connected to the OBD-II port. All data were collected from a single vehicle,

which was on a dynamometer during the attacks. Targeted ID injection attacks were performed using the flam delivery technique, where a message is injected immediately after a legitimate message with the target ID is seen. These attacks assume the attacker has white box knowledge, achieved using a signal reverse engineering algorithm.

Despite its various limitations, the HCRL CH dataset is the most commonly used CAN bus attack dataset for evaluating CAN ID-based IDSs. To compare the CAN-CID with baseline models and to assess its generalization capability, this dataset is utilized as one of the evaluation datasets. The HCRL CH dataset was created by logging CAN traffic via the OBD-II port from a real vehicle while message injection attacks were performed. However, further details about the threat model are not discussed in their documentation. The HCRL SA dataset, released by the same authors of the HCRL CH dataset, includes data from three vehicles, making it a valuable dataset for evaluating the generalization capability of IDSs. Therefore, this dataset is also utilized to evaluate the CAN-CID model. Like the HCRL CH, the HCRL SA dataset was collected by logging CAN traffic via the OBD-II port from three real vehicle brands. It includes flooding, fuzzing, and malfunction attacks. These HCRL datasets are not used to evaluate the CAN payload-based IDS due to their limited sizes. The SynCAN dataset is designed to provide a viable dataset for training unsupervised signal-based CAN IDS. It is widely used in the literature for evaluating unsupervised CAN payload-based IDS. Consequently, along with the ROAD dataset, the synthetic SynCAN dataset is employed to evaluate the Latent AE model. The SynCAN dataset simulates two injection and three masquerade attacks.

Initial experiments for the CAN-ID and payload-based models utilized publicly available datasets, including ROAD, HCRL CH, HCRL SA, and synCAN. These datasets, however, have various limitations. For instance, the HCRL datasets only include two driving behaviours during benign and attack data collection, the ROAD dataset's attack data collection did not involve active driving on a road, and the SynCAN dataset's synthetic nature. To address these limitations, a novel CAN bus attack and benign dataset is collected and introduced in Chapter 8. This was conducted during the final six months of this research for two main reasons. First, the proposed models needed to be evaluated with multiple publicly available benchmark datasets to fine-tune them and improve their generalization capabilities. These experiments highlighted both the well-known limitations of the benchmark datasets and significant less-known limitations, such as the lack of focus on medium and low-frequency IDs of the ROAD dataset attacks. These findings helped create a more comprehensive dataset for thorough IDS evaluation. Second, creating a comprehensive dataset requires technical knowledge, time, and resources. To this

end, we collaborated with our industry partner Horiba MIRA, and our experiments had to align with their proving ground availability. Using a single vehicle, both benign and attack datasets were created to simulate real-world driving conditions. Assuming the grey-box knowledge of the attacker, specific sensor values for some IDs were manipulated based on the vehicle's CAN DBC file, while random injections were employed for others. Labels were added during data pre-processing to facilitate IDS evaluation. This dataset is utilized to evaluate the deployed model discussed in Chapter 9. Further details of the data preprocessing techniques employed, such as cleaning, feature extraction, and normalization, can be found in each chapter discussed in the rest of the thesis.

4.4 Research Ethics

Prior to the beginning of this research work, the "Research Ethics: Research Student and Supervisor Assessment (RESSA)" form was completed to adhere to ethical procedures. This research does not involve or disclose information related to individual human subjects, groups, organizations, animals, or genetically modified organisms. The data used are related to vehicle networks. The RESSA form was reviewed frequently at various stages of the research to ensure compliance with ethical procedures. The models proposed in Chapter 5, Chapter 6, and Chapter 7 utilize benchmark datasets, and therefore, these works pose a negligible level of risk in terms of ethics and infrastructure.

Model deployment, discussed in Chapter 9, involves both humans and vehicles. Therefore, safety protocols were strictly followed during the data collection and model deployment. These protocols were based on the MIRA health and safety procedure MN2145/S/03 and the associated risk assessment procedure. MN2145/S/03 includes a vehicle test activity safety checklist. According to this procedure, the method statement specifies that:

- The vehicle will be manually driven on the Proving Ground's exclusive facility, with the speed limited to 30 mph.
- The driver will be responsible for the vehicle and will constantly monitor the environment, assessing the safety of the tests and the vehicle.
- The researcher will sit in the back seat and deploy the models using a laptop connected to the vehicle's CAN bus.

MN2145/S/03 identifies potential hazards and controls appropriate for the test, fully assessing the risk. It mandates the use of standard seatbelts, fire extinguishers, and

additional safety measures. The risk assessment procedure includes criteria such as the task, hazard type, who or what might be harmed and how, current control measures, and further control measures. According to this procedure, all current control measures are adequate for the data collection experiments. The risk assessment procedure used is shown in Table 4.1 and Table 4.2. Consequently, the MIRA health and safety procedure and risk assessment procedure were rigorously followed during the data collection and model deployment.

Based on our observations during the attack period, we adhered to responsible disclosure protocols and ethically reported all findings, including vulnerabilities, to MIRA management. The vehicle used in the experiments was modified by MIRA, and some of these changes led to vulnerabilities, such as reduced steering control during the attack period. As responsible researchers, we shared all collected data, observations, and reports with MIRA and obtained their approval before publishing to prevent any disclosure that could put the sector at an unfair disadvantage.

4.5 Assumptions and Limitations

Several assumptions are made during the development of models. For example, the CAN ID-based model discussed in Chapter 5 and Chapter 6 assumes that the number of CAN IDs of a vehicle is fixed. This assumption is derived from the analysis of CAN data from multiple vehicles using benchmark datasets. The Latent AE model assumes that the minimum and maximum values for each variable are observed in the training dataset for model training. Both ID and payload-based IDS assume that anomalies are attacks, as it is challenging to distinguish between benign anomalies and cyberattacks. The proposed models and model deployment have several limitations, such as overlooking some variables in the Latent AE model and evaluating IDS performance using only one vehicle during model deployment. These assumptions and limitations are discussed in detail in each chapter.

Table 4.1: MIRA risk assessment procedure - Part 1

Task	**Hazard type**	**Who, what might be harmed and how?**	**Current control measures**
Driving and any work within or near the vehicle	Fire	Vehicle occupants, pedestrians, and other road users	The powertrain and the high voltage systems have not been modified and remain in the original OEM state. Risk Assessment in place. All work is limited to low risk activities.
Manual Driving	Single vehicle control loss and multiple vehicle interaction	Vehicle occupants, and other road users	The Sygnal Drive By Wire (DBW) system has to be disabled at all times when driving on public roads or on the Proving Ground. This is achieved by ensuring the e-Stop button is pressed. The DBW system is engineered to strict safety standards.
Manual Driving	Single vehicle control loss and multiple vehicle interaction	Torque applied to the hand wheel by the electric power steering motor, causing the driver to lose control, leading to a collision that harms occupants of test vehicle and other vehicles as well as damaging the vehicles and any other objects involved in the collision.	The Sygnal Drive By Wire (DBW) system has to be disabled at all times when driving on public roads or on the Proving Ground. This is achieved by ensuring the e-Stop button is pressed. The DBW system is engineered to strict safety standards. The driver's driving licence verified by HR to ensure they are competent to drive on and off site.
Manual Driving	Single vehicle control loss and multiple vehicle interaction	Driver is distracted by viewing or operating laptop, leading to a collision that harms occupants of test vehicle as well as damaging the vehicle and any other objects involved in the collision.	The driver's sole task will be to operate and monitor the vehicle. Other tasks will be conducted by the laptop user on the back seat. The laptop shall be located in the back seat and not observed by the driver.
Manual Driving	Proving Ground use	Proving Ground users	All driving will be contained within the designated area as approved by the Assured Cav Team. Vehicle speed shall be limited to 30mph.
Manual Driving	Proving Ground use	Collision with stationary obstacle resulting in harm to the occupants of the test vehicle and other vehicles.	Driver to adhere to the Highway Code at all times. Driver to be vigilant while driving on the Proving Ground. All driving will be contained in the designated area as approved by the Assured Cav Team. Vehicle speed shall be limited to 30mph. The driver's driving licence verified by HR to ensure they are competent to drive on and off site.
Manual Driving	Vehicle collision	Risk of injury from collision with another road vehicle, cyclist, pedestrian, and road furniture	Driver to adhere to the Highway Code at all times. Driver to be vigilant near pedestrian crossings and junctions where cycles and pedestrians may cross. The driver's driving licence verified by HR to ensure they are competent to drive on and off site. Vehicle is maintained, serviced and has appropriate MOT to ensure it is roadworthy
Manual Driving	Parking Manoeuvres	Risk of collision with pedestrians, cyclist, other vehicle or property during parking manoeuvres	All road markings, such as double yellow lanes, should be adhered to and the vehicle should not be parked where these markings exist, or where other restrictions are in place, or where they obstruct other vehicles or access. Company reverse parking policy to be observed when parking on site. Driver must pay attention when reversing to other traffic or pedestrians around them.

Table 4.2: MIRA risk assessment procedure - Part 2

Task	**Hazard type**	**Who, what might be harmed and how?**	**Current control measures**
Manual Driving	Adverse Weather	Vehicle Occupants, Other Road users, Buildings or Road Furniture Users, Pedestrians	Driver is encouraged to drive only if driving is necessary in bad weather conditions. Driver must adhere to speed limits and be extra vigilant of speed limits and manoeuvres during bad weather conditions. Driver must use fog lights during extreme foggy conditions to alert other road users of their presence.
Manual Driving	Projectiles	Large accelerations in any direction cause the laptop to hit and harm test vehicle occupants as well as damaging the laptop and parts of the vehicle interior.	The individual responsible for operating the laptop will ensure that the laptop is secured to an appropriate restraining device or controlled by operational procedure.
Manual Driving	Accident due to driver error	Vehicle occupants, pedestrians, and other road users,	Team members driving licences verified by HR to ensure they are competent to drive on and off site. Vehicle is maintained, serviced and has appropriate MOT to ensure it is roadworthy. Driver to take adequate rest breaks to avoid tiredness. Driver advised not to drive if feeling tired, unwell or unfit to safely control the vehicle.
Manual Driving	Other Road Users	Collision with other vehicles, pedestrians and test equipment located within the designated area of the Proving Ground leading to injury and/or damage to test equipment and vehicles.	Trained driver will be informed of the vehicle and will be briefed on which area of the Proving Ground to access. Driver is encouraged to drive only if driving is necessary in bad weather conditions. Driver to adhere to the Highway Code at all times. Driver to be vigilant while driving on the Proving Ground. All driving will be contained in the designated area as approved by the Assured Cav Team.
Data Collection	Access/Egress	The cables connecting the CAN data device (P-CAN) to the OBD port and the laptop can cause a trip hazard when entering or exiting the vehicle	The data collection device will be securely connected the OBD- Port and all cables will be secured inside the vehicle. The test equipment will be installed and secured to the vehicle in the lab before the testing exercise. No data analysis will be conducted in the vehicle, all analysis will be conducted in the lab.
Test Equipment	Projectiles	PEAK System (P-CAN, Cables, laptop) may be thrown or projected as a result of acceleration or deacceleration	The PCAN and its cables will be secured when testing in the lab before testing. The laptop will be secured in the back of the vehicle and will not be interacted with when driving.

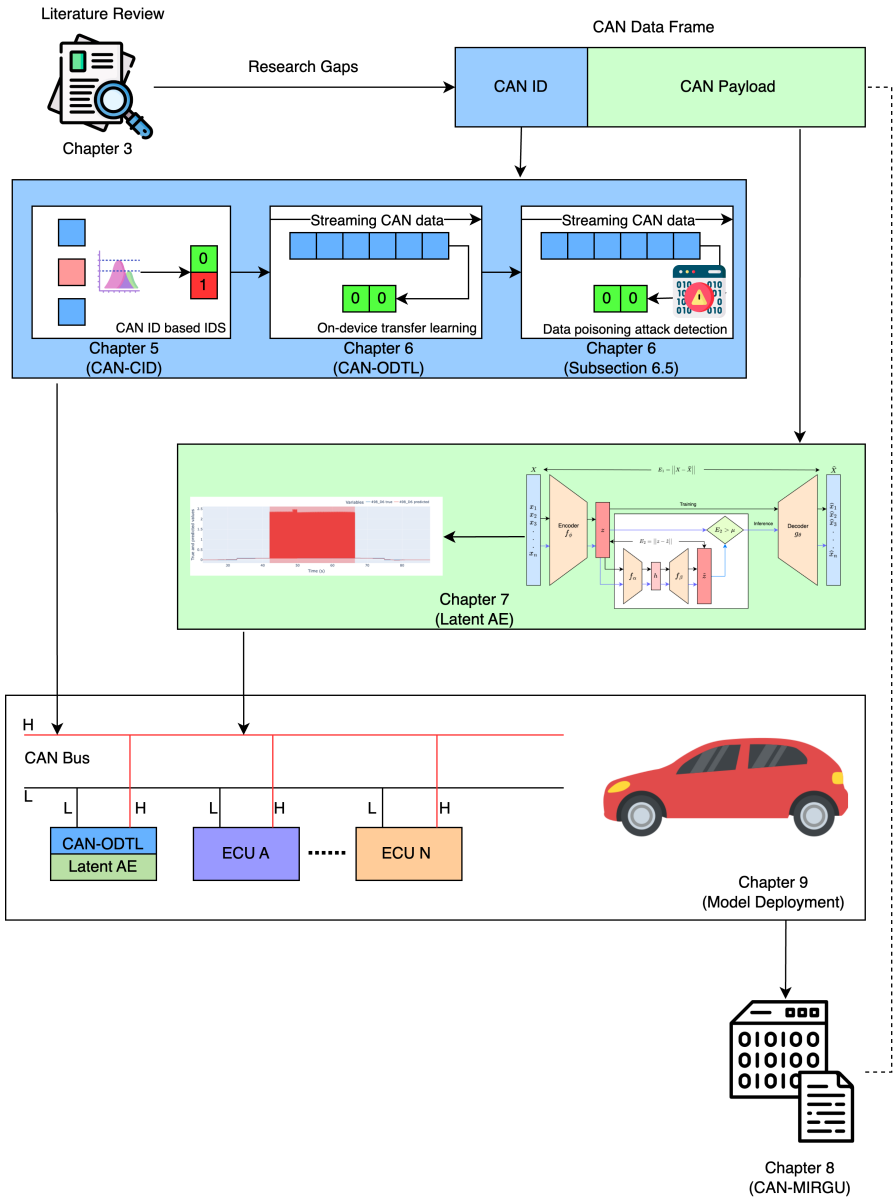


Figure 4.1: Overall design of the Thesis

Chapter 5

Context-aware CAN ID-based Intrusion Detection System

Based on the literature review in Chapter 3, it is clear that existing IDSs developed to detect cyberattacks on IVNs have many limitations, such as higher detection latency and low detection accuracy. However, many works have proved that CAN ID can be used to detect attacks on the CAN bus, specifically injection attacks. Additionally, a few works have used time as a feature to improve the detection of injection attacks. Drawing inspiration from these works and considering the limitations of the proposed solutions, this chapter proposes a context-aware CAN ID-based IDS utilising the frequency behaviour of CAN IDs. The proposed lightweight solution, designed considering the resource constraints on IVNs, shows that it can detect a wide variety of attacks with low detection latency. This chapter addresses the RQ2.

The main findings of this chapter were presented at the 14th International Conference on Cyber Conflict: Keep Moving (CyCon) 2022 [26]

5.1 Introduction

Developing an in-vehicle IDS for widespread adoption with high detection capability encounters challenges, including lack of knowledge about CAN data specifications, high-frequency data transmission, computationally constrained in-vehicle environments, and various attacks altering different data fields of CAN messages to compromise the in-vehicle network [30]. Moreover, numerous events in a vehicle might be deemed anomalies,

even if they align with legitimate driving scenarios. For example, emergency braking or a sudden steering wheel turn at 70 mph could be considered anomalous under normal driving conditions. These benign anomalous behaviours can lead to a significant number of false positives.

Frequency or time-based IDSs proposed in the literature utilise the timing of CAN frames or the sequential nature of the IDs. These IDSs are discussed in Section 3.3.1 in detail. Accordingly, in [58], a context-aware anomaly detector for monitoring cyberattacks on the CAN bus was developed, utilising sequence modelling. Additionally, in [59], the authors proposed an anomaly detection algorithm by modelling the normal behaviour of the CAN bus, considering the recurring pattern of CAN IDs. This is equivalent to 2-grams in the N-gram-based model used in [58]. While N-gram-based algorithms capture context, they often incur high computational overhead as N increases. In [188], the authors introduced a time-based IDS to detect CAN injection attacks, achieving 98.6% precision for selected injection attacks. However, they argued that time-based models alone are insufficient, leading to a high rate of false positives per minute due to the frequent data transmission. Another approach proposed by [69] employed a rule-based and supervised LSTM model in an ensemble framework for CAN bus attack detection, outperforming individual models. Despite the superior detection capability demonstrated by supervised deep learning-based IDSs, they may have low generalization capability to other attacks and vehicles, as they learn specific patterns from the training dataset. Additionally, their complex architectures contribute to high detection latency.

To effectively address the challenges and limitations encountered by the existing IDSs in the literature, this chapter introduces CAN-CID (CAN Centre ID prediction), a novel context-aware ensemble IDS for the CAN bus. The proposed system leverages natural language processing (NLP) and time-based techniques to enhance its capabilities. The CAN ID field is comparatively simpler than the multivariate payload field. Certain attacks, like injection attacks, can substantially alter the ID field during an attack. Therefore, CAN ID-based IDSs are effective in detecting such attacks without requiring extensive assumptions or prior knowledge of CAN data specifications.

5.2 Chapter Contribution

The main contribution of this chapter can be summarized as follows:

1. CAN-CID uses only benign data to train the model and estimate thresholds. This

avoids the need to collect real attack data to train the algorithm. It is significantly easier and safer to collect benign CAN data from real vehicles than to collect attack data. Further, using only benign data (one-class) during the training process improves the generalization capability of the algorithm.

2. Probability-based thresholds were estimated for each ID using only benign training data. The minimum softmax probabilities for the benign data of each CAN ID were selected as the thresholds to minimize false positives, thereby enhancing the overall accuracy of the ensemble model.

3. CAN-CID uses a one-layer shallow GRU network to detect anomalous ID sequences. Hence, it is lightweight, and detection latency is very low (10 ms for a 100 ms window). This makes the proposed solution suitable for deployment in real vehicles.

5.3 The Proposed CAN-ID based IDS

This section introduces the CAN-ID based IDS, leveraging the sequential properties inherent in the CAN ID streams.

5.3.1 CAN Centre ID prediction

This work is inspired by the approaches presented in [58] and the continuous bag-of-words (CBOW) model architecture proposed by [189]. In [58], the authors utilised N-gram distributions to construct a CAN ID sequence model. The fundamental concept underlying this work involves a mathematical model (n-gram) that can be trained to learn CAN message sequences and predict subsequent elements in the sequence. The authors demonstrated that event occurrence (ID) could be determined based on a short history. However, this dependency might scale with the number of nodes in the network (equivalent to the number of unique words in a language). Therefore, for a larger number of nodes, a longer history may be necessary, as a larger number of unique sequences could be generated for the selected window size. N-gram models become inefficient for higher values of N due to the increased number of combinations. This approach [58] shares similarities with the next-word prediction task in natural language processing (NLP), where predictions are made based on the preceding words (context). The Word2vec model, introduced by [189], focuses on learning word vectors (word embeddings) by predicting the centre (target) word given the context. The architecture of this model is illustrated in Figure 5.1. The CBOW model is designed to learn word vectors that

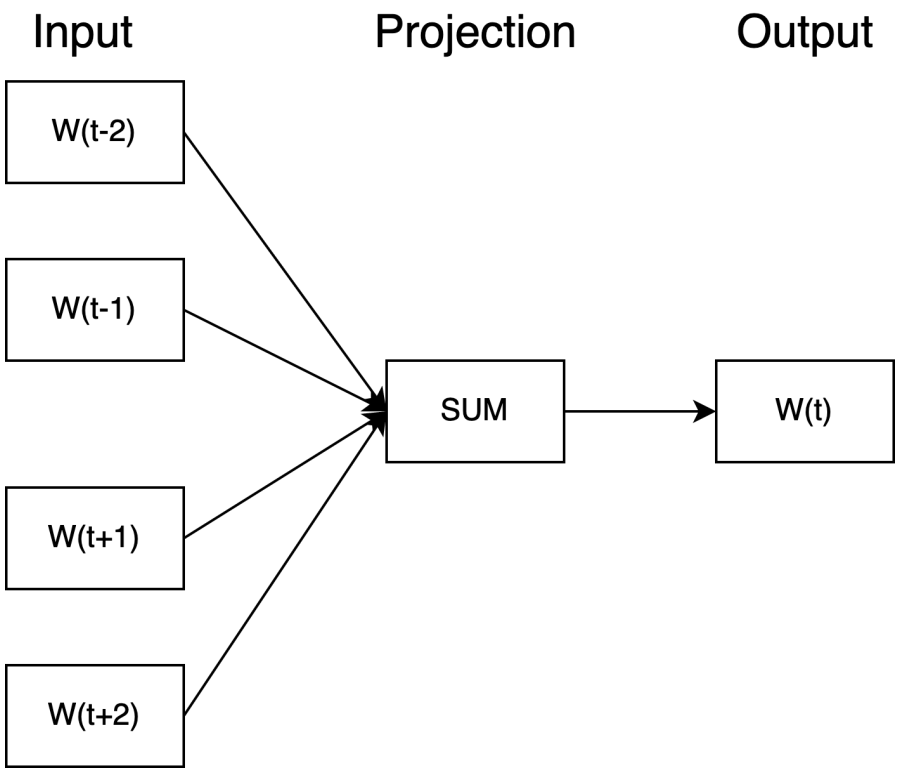


Figure 5.1: Continuous bag-of-words (CBOW) architecture to predict the centre word given the previous and next words as the context

represent the meaning of the middle word and its context. While the CBOW model's primary objective is not word prediction, it aims to learn accurate word vectors that encode semantic relationships for all words in the corpus. Subsequently, these learned word vectors can be applied in various language models employing specific deep learning architectures.

Using the target word's historical (pre-context) and future words (post-context) as the input words improved the centre word prediction[189]. We expect the same behaviour for CAN ID sequences. To elaborate on this, consider a driving scenario of a right-hand turn at an intersection. Possible events in the vehicle include activating signal lights, decelerating (applying brakes), stopping, accelerating, and turning right. If we want to predict the third event, which is stopping in this case, we can use only previous events as the context or use both previous and future events as the context. These two tasks can be formulated as follows:

$$x_1 = \{\text{activate signal lights, decelerate}\}, \quad y = \{\text{stop}\} \tag{5.1}$$

$$x_2 = \{\text{activate signal lights, decelerate, accelerate, turn right}\}, \quad y = \{\text{stop}\} \tag{5.2}$$

The second task can enhance the accuracy of predicting 'stop' as the number of possible events for the centre event will be equal to or fewer compared to the first task. For instance, in Equation 5.1, 'accelerate' might be another likely prediction. However, in the second task, having 'accelerate' in the context makes the prediction of 'stop' more precise. Therefore, the CBOW architecture is employed to deduce the context for CAN ID sequences. One limitation of the CBOW approach is that it needs to wait for a few messages to determine if the target ID is malicious. Nevertheless, considering the CAN ID transmission rate, this delay will be minimal (approximately a 0.005 s delay for 10 IDs). Hence, CBOW stands as a viable option for detecting attacks in CAN ID sequences.

5.3.2 CAN-CID Architecture

Figure 5.2 illustrates the structure of the proposed model. A sliding window with a size of n (number of IDs) is implemented within a larger sliding window with a size of T (time), where n is an odd number. Let N denote the total number of unique CAN IDs. In the GRU-based model, an embedding layer is employed as the initial layer to learn accurate word vectors that encode semantic relationships for all the IDs in the CAN bus. Consequently, the input to the embedding layer comprises a sequence of vectorized CAN IDs of size $(n-1)$. The centre (middle) ID $(n+1)/2$ serves as the target for prediction. utilising a single GRU layer as the hidden layer enables the learning of temporal patterns. A dropout layer is incorporated to mitigate overfitting and enhance model generalization. Finally, a dense layer of size N is employed as the classification layer with the softmax activation function to obtain the probability for each CAN ID. During the training, the overall objective of this model is to learn to predict the centre ID given the $(n-1)/2$ pre and post-context. This can be achieved by minimizing the objective function of categorical cross-entropy. This is given by:

$$E = -\sum_{i=1}^{N} y_i log(\hat{y}_i) \tag{5.3}$$

where y_i is the true label, and $\hat{y}_i$ is the predicted softmax probability for the i^{th} class. Parameters W_1, W_2, and W_3 are learned using backpropagation during the model training. The GRU model is trained with a sufficiently large benign dataset to learn benign sequences. This approach can be viewed as a form of self-supervised learning, as it uses

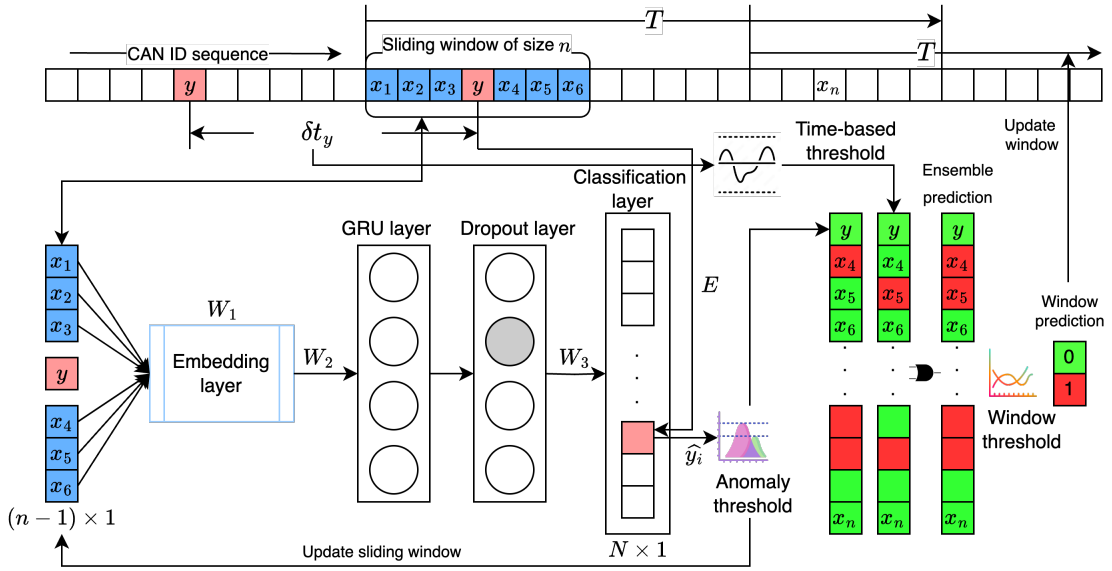


Figure 5.2: Ensemble model architecture

the input data itself for supervision [190]. During inference, the predicted softmax probability $\hat{y}_i$ for the target ID (y) is then compared with the pre-defined anomaly threshold. If the predicted probability falls below the threshold, the target ID is flagged as a weak anomaly; otherwise, it is identified as a benign ID. Similarly, the time-based model assesses the time between two consecutive target IDs (δt_y) against pre-defined time-based thresholds (minimum and maximum time). If δt_y is below the minimum or exceeds the maximum time thresholds, the current ID is marked as a weak anomaly; otherwise, it is labelled as a benign ID. This process extends to all IDs within the window T. The OR operator combines both models into an ensemble model. Ultimately, the window threshold is applied to categorise the time window T as a malicious or benign sequence. This procedure repeats for all IDs in the CAN ID stream by sliding the time window T, with an overlap of $(n - 1)$ IDs to predict the missing IDs from the preceding window.

This procedure is shown in Algorithm 1. Firstly, it extracts the context IDs (x), centre ID (y), and associated timestamp (t) from streaming CAN data (F). The minimum time (t_min) within each window is utilised to identify frames belonging to the time window T. Using the trained GRU-based model (M), softmax probability for the centre ID is predicted. Then, the inter-arrival time for the centre ID is calculated (δt_y). If the predicted softmax probability for the centre ID is less than a pre-defined threshold

of ω or the inter-arrival time is below the minimum (t_{min}) or exceeds the maximum time (t_{max}) thresholds, the frame is declared as a weak anomaly. These anomalous and benign frames in the observation window are used to detect the window as anomalous or benign based on a window threshold of ψ.

Algorithm 1 CAN GRU and Time-based ensemble anomaly detection

Input: Streaming CAN data F, Anomaly threshold ω, Window threshold ψ, Time window T, Time-based thresholds t_{min}, t_{max}, Trained model M,
Output: Anomaly status for each window

while F is not empty **do**
  read x, y, time_stamp t, t_min
  while $t - t\_min \leq T$ **do**
    Init: Benign count $C_b = 0$, Anomaly count $C_a = 0$
    $\hat{y} = M(x)$
    Compute δt_y
    if $\hat{y} < \omega$ or $t_{min} > \delta t_y > t_{max}$ **then** ▷ for id y
      Declare y as a weak anomaly
      $C_a = C_a + 1$
    else
      Declare y as a benign
      $C_b = C_b + 1$
    end if
  end while
  if $C_a/(C_a + C_b) > \psi$ **then**
    Return Anomaly
  else
    Return Benign
  end if
  $t\_min \leftarrow t$
end while

5.3.3 Threshold Estimation

The proposed model employs three thresholds: anomaly, time-based, and window thresholds. Since the model training process is entirely based on benign data, all thresholds must be derived from benign datasets. Thus, a separate benign dataset is used to estimate the anomaly and window thresholds, while the training data is employed to determine the time-based thresholds.

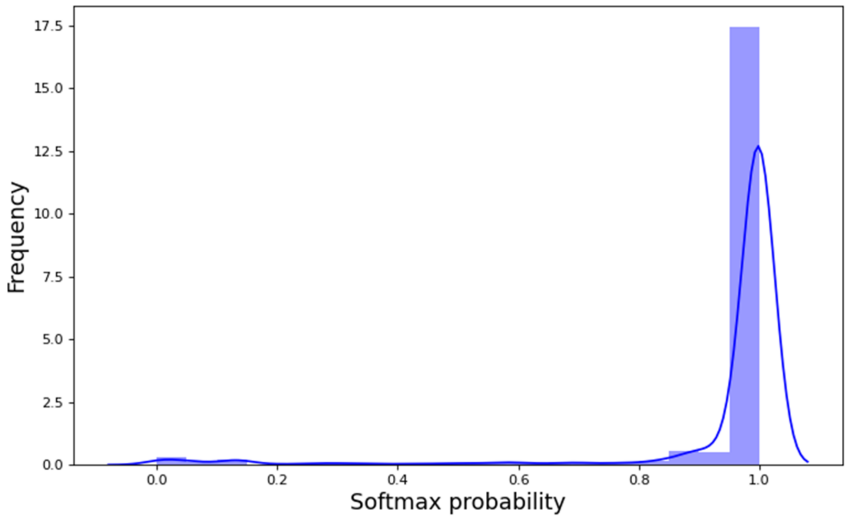


Figure 5.3: Softmax probability distribution of ID 580

Anomaly Threshold

To determine the anomaly threshold (ω), softmax probabilities were computed for all IDs in the benign sample using the trained GRU-based model. While the minimum softmax probability values of each ID are preferred as threshold values to minimize false positives, there may be unseen benign sequences in the threshold estimation benign dataset that were not observed in the training dataset. Consequently, these benign frames tend to have lower probabilities. Therefore, we consider the N^{th} lowest quantile values as the anomaly thresholds. Thus, any value below these selected minimum thresholds is regarded as anomalous. Figure 5.3 illustrates a softmax probability distribution for a specific selected ID.

Time-based Threshold

To establish the time-based threshold, the training dataset was utilised, as it does not involve a separate training phase. For each ID, the time difference between two consecutive frames within the benign dataset was computed. As demonstrated in previous studies [188], since time-based models are prone to false positives, to minimize false positives in the ensemble model, the minimum and maximum time difference values between two consecutive frames for each ID were then employed as the corresponding thresholds. The minimum values detect injection attacks, while the maximum values identify suspension attacks, as injection decreases the message inter-arrival time, and suspension

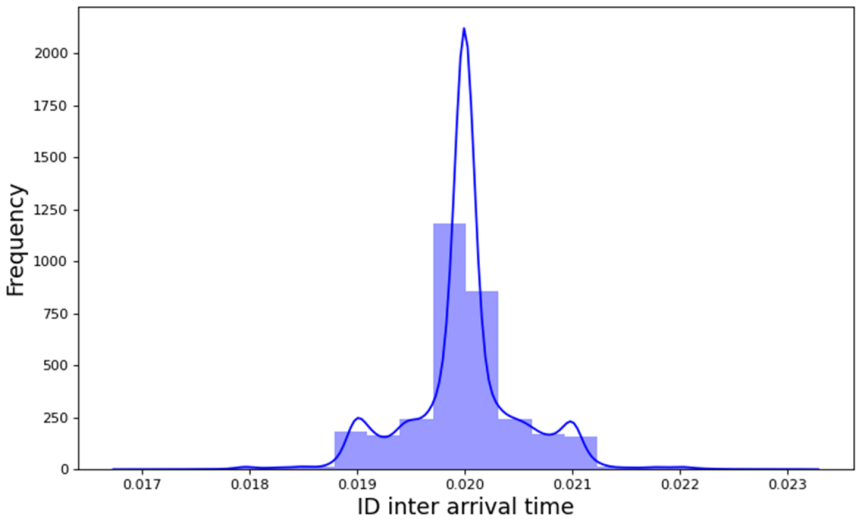


Figure 5.4: Inter-arrival time distribution for ID 580

increases it. Figure 5.4 illustrates the distribution of inter-arrival times for a selected ID. Despite transmitting IDs based on a predefined frequency of 0.02s, the presence of an ID-based priority mechanism in the CAN bus, and the randomness incurred from jitters [71], result in a Gaussian distribution with minor variations. This behaviour leads to the generation of a large number of CAN ID sequences for a fixed window size.

Window Threshold

We employed ID-based and time-based thresholds for detecting weak anomalies. Counting weak anomalies over a window (T) aids in minimizing false positives [58]. Therefore, we established a window threshold (ψ) to classify windows as either attack or benign. The ensemble model is used to determine the anomalous or benign status of each frame in a benign dataset using the defined anomaly and time-based thresholds. The average false positive rate is then computed for each time window T, and this value is used to set the window threshold. For example, if benign windows produce an average of two false positives, then at least three anomalous frames should be present to consider the window status as anomalous. This helps to reduce the false positives of the proposed model. Labels were assigned to each window (0 for benign and 1 for attack) as the ground truth if at least one attack frame is presented in the window, and this information was used to assess the model's performance. It is worth noting that the GRU-based model may identify several frames, besides the actual injected frame, as weak anomalies because the injected frame might create several new (anomalous) CAN ID sequences.

5.4 Evaluation and Performance Results

This section introduces the dataset employed in the experiments, outlines the experimental setup, and presents the results for the proposed model.

5.4.1 Threat Model and Datasets

In this study, the proposed model was tested using the ROAD CAN intrusion dataset [30]. To assess the model's generalization capability, two other publicly available datasets, HCRL CH [72] and HCRL SA [126], were also employed. Descriptions of these datasets can be found in Section 3.5. The ROAD dataset is considered the first open CAN bus dataset with advanced types of real attacks that have physically verified effects on the vehicle [30]. It encompasses 12 ambient (benign) datasets, each representing various driving activities such as driving, accelerating, decelerating, reversing, braking, cruise control, and the activation of turn signals. Additionally, it includes anomalous yet benign driving activities like unbuckling a seatbelt and opening doors while driving. Detailed information about these benign datasets is provided in Table 5.1. The chosen attacks, as outlined in Table 5.2, were specifically selected to evaluate the algorithm's detection capabilities. In these targeted ID attacks, a frame with a specific ID is injected immediately after the appearance of a legitimate frame. The objective is to prompt the vehicle to disregard the legitimate message and accept the injected frame, thereby altering the vehicle's state. The HCRL CH dataset includes DoS, fuzzy and spoofing (RPM and gear) attacks, whereas the HCRL SA dataset includes flooding, fuzzy and malfunction (targeted ID) attacks.

5.4.2 Experimental Setup

Since the dataset includes the different contexts of benign driving behaviours, first, each benign dataset listed in Table 5.1 except the basic short, splits into two parts: training (70%) and threshold estimation (30%) while preserving the temporal behaviour of the CAN data. We assume that, for each benign dataset, both the training and threshold estimation segments exhibit similar driving behaviours, resulting in comparable distributions. Subsequently, the training splits are combined into a unified dataset for model training, while the threshold estimation splits are combined into a distinct dataset for the purpose of estimating anomaly thresholds. The basic short dataset is specifically employed as the representative sample for estimating the window threshold. The selection of a representative sample for threshold estimation is crucial, as the false positive and

Table 5.1: Description of ROAD benign datasets

Dataset	**Driving activities**	**Driving time (s)**
Basic long	Basic driving activities	1250
Basic short	Basic driving activities	444
Reverse	Basic reverse activities	51
Benign anomaly	Driving while trying to cause benign anomalies	456
Extended long	Basic driving activities and more complex/one-off activities	657
Extended short	Basic driving activities and more complex/one-off	359
Radio infotainment	Playing with radio and infotainment unit while idling and driving	390
Idle radio infotainment	Playing with radio and infotainment unit during while idling	660
Drive winter	Driving and accelerating in colder conditions	47
Exercise all bits	Trying to exercise full range of all signals	2172
Highway street driving	Drive in parking lots, city streets and highways	469
Highway street driving long	Drive in parking lots, city streets and highways	3764

Table 5.2: high-frequency injection (fabrication) attacks on the road dataset

Attack	**Attack technique**	**Consequence**
Fuzzing	Inject random IDs and arbitrary payloads	Wide variety of unexpected results
Correlated signal	Inject false wheel speed values (ID -6E0)	Stop the car due to different pairwise wheel speeds
Max speedometer	Change one byte of payload to maximum (FF) value (ID-0D0)	Display false speedometer value
Reverse light on attack	Change one bit of payload (ID-0D0)	Reverse lights do not reflect what gear the car is using
Reverse light off attack	Change one bit of payload (ID-0D0)	Reverse lights do not reflect what gear the car is using

negative rates are highly dependent on the chosen anomaly thresholds.

To create the sliding window, we selected five IDs from each side of the target ID ($n = 10$). Larger window sizes increased model complexity, while smaller ones led to decreased accuracy in predicting the center IDs. Following extensive experimentation with varying window sizes, we selected the optimal window size that strikes a balance between model complexity and average center ID prediction accuracy. Additionally, a time window of 100ms was selected, representing smaller windows suitable for near-real-time detection. This selection aimed to achieve a balance between minimizing false positives and meeting the need for near real-time detection. This configuration resulted in approximately 250 IDs per prediction window. To enhance the model's efficiency, a hidden layer with only 32 GRU nodes was employed, followed by a 0.2 dropout layer. The datasets, namely ROAD, HCRL CH, and HCRL SA, consist of 106, 27, and 45 nodes, respectively (N). For the ROAD dataset, the window threshold was set to 0.01, while for the HCRL datasets, the

threshold was set to 0.1 based on the observed average anomalies in the benign datasets. Allowing a small margin for unseen benign data, 0.001th quantile values were used for the anomaly thresholds in all datasets. Opting for minimum values for these thresholds is advantageous in minimizing false positives. However, these minimal values could arise from unseen benign sequences. Hence, through repeated experiments, we determined the 0.001th quantile value to be the suitable threshold. Hyperparameter optimization was performed using a grid search, and the same parameters determined for the ROAD dataset were applied to both HCRL datasets. We selected the smallest and most optimal hyperparameters for n, the number of GRU nodes, and the embedding size. The proposed algorithm was implemented using Python 3.8 with TensorFlow and the Keras library. All experiments were conducted on a MacBook Pro 2.2 GHz Intel Core i7 with 16 GB RAM.

We compared CAN-CID with two baseline methods, that is, the N-gram-based model (N-gram) [58] and the transition matrix-based model (transition matrix)[59], where both baseline models identify anomalies based on observed benign ID sequences. Additionally, we introduced a variant of CAN-CID known as CAN-NID (CAN Next ID prediction). CAN-NID shares similarities with CAN-CID, except that the GRU model considers context IDs from one side (previous IDs). Optimized hyperparameters for the CAN-NID model include 16 previous IDs as the context, two hidden GRU layers with 128 nodes, and a dense output layer with a softmax activation function. We conducted fine-tuning for both baseline models on each dataset to ensure a fair comparison with our model. To evaluate the model performance, we used F1-score, false-positive rate (FPR) and false-negative rate (FNR).

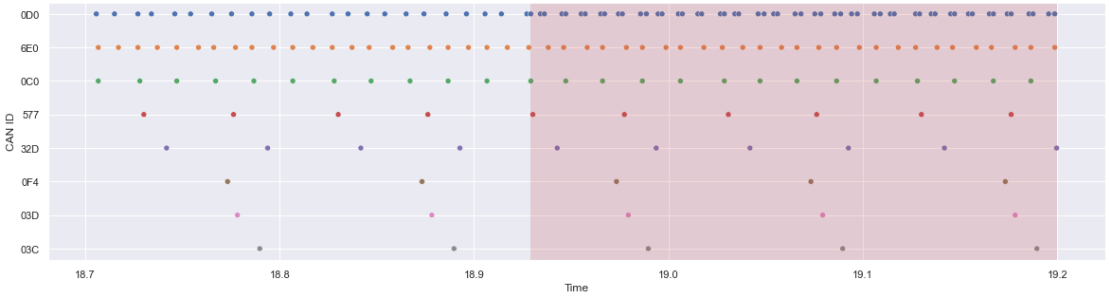
5.4.3 CAN ID Data Analysis

We performed an analysis of CAN ID data to differentiate between benign and anomalous frame transmission patterns, identify the patterns of benign ID sequences, and underscore the significance of using both pre and post-context to predict a centre ID. To achieve this, we utilised the samples of benign and attack data of ROAD CAN Intrusion dataset [30].

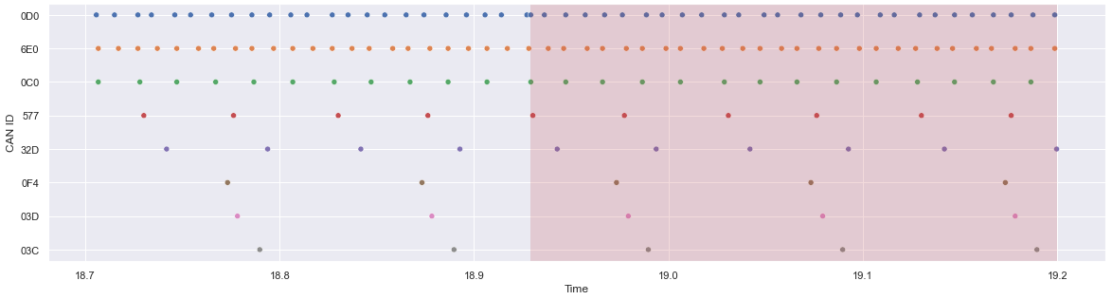
CAN Frame Transmission Patterns

In Figure 5.5a, a targeted ID attack focusing on the ID 0D0 is depicted within a five-second snapshot, emphasizing the periodic nature of the IDs. The nodes (ECUs) consistently exhibit frame transmissions at fixed intervals, aligning with observations in [9]. While 104 out of 106 IDs in the ROAD dataset display similar frequent behaviour, the injected ID disrupts this pattern. Figure 5.5b illustrates a five-second snapshot of a

masquerade attack for the same ID (0D0), indicating that masquerade attacks may not significantly alter the ID transmission frequency [30]. However, a masquerade attack might cause a slight deviation (shift of time) in the frame transmission time due to the difficulty of time synchronization with the legitimate ECU [71]. Additionally, since a masquerade attack stops the frame transmission of a legitimate ECU, there might be a brief period where no frame is transmitted with the targeted ID. Given the high message transmission rate on the CAN bus, even a minor deviation from the normal driving scenario could generate new ID sequences. For instance, ID 0D0 might have the sequence '0D0 6E0 0C0' during normal driving, while a slight time shift or absence of frames could create a new sequence like '6E0 0D0 0C0'. This behaviour (frequency and sequence change) can be observed for all injection and masquerade attacks in the ROAD dataset.



(a) Frame transmission of a targeted ID (0D0) attack



(b) Frame transmission of a masquerade attack (0D0)

Figure 5.5: Frame transmission of ID 0D0. The shaded area represents the attack period. this represents only a subset of the 106 CAN IDs

CAN ID Sequences

To identify the patterns of ID sequences within a fixed window size, we utilised a sample of benign data from a 30-minute drive. Figure 5.6 illustrates the frequency of the top 20 sequences for five consecutive IDs (window size). This reveals that certain sequences

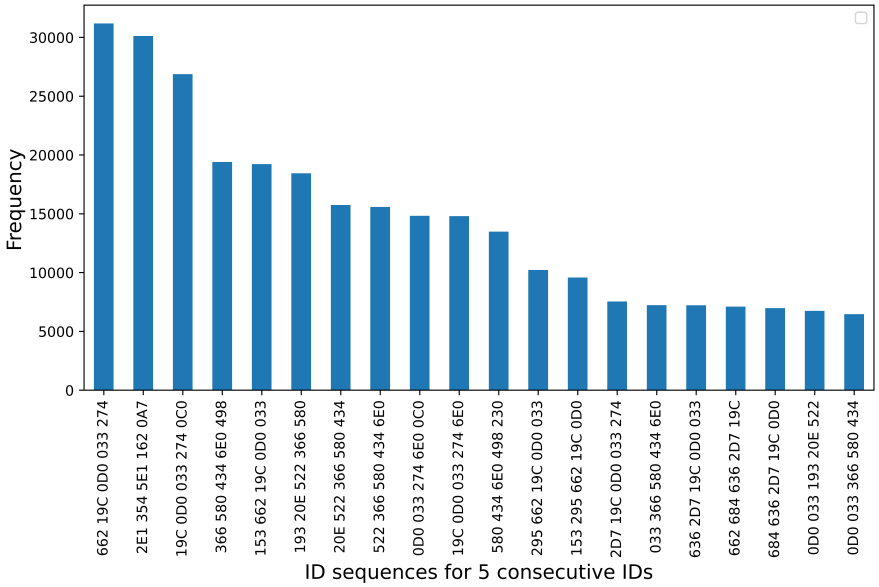


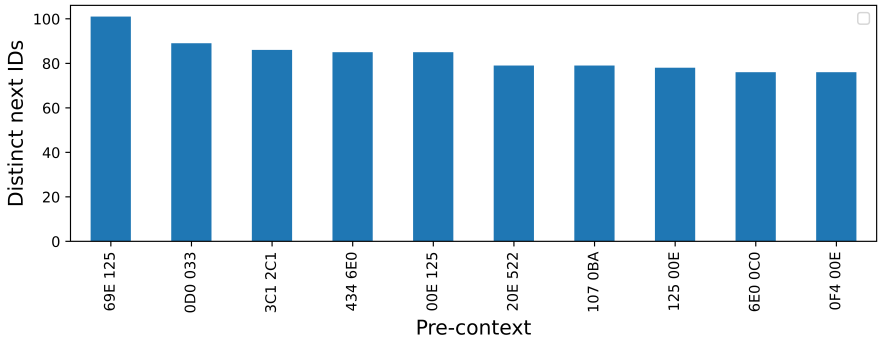
Figure 5.6: The top 20 CAN ID sequences for five consecutive IDs

occur with higher frequency, while others exhibit lower frequency. These patterns can be leveraged to predict the subsequent CAN IDs. For example, given the context '662 19C 0D0 033', the model can predict that ID 274 is the most likely next CAN ID. During an injection attack, new CAN ID sequences may emerge, which are not typical during normal driving periods. This is mainly because new frames appear in an unusual context.

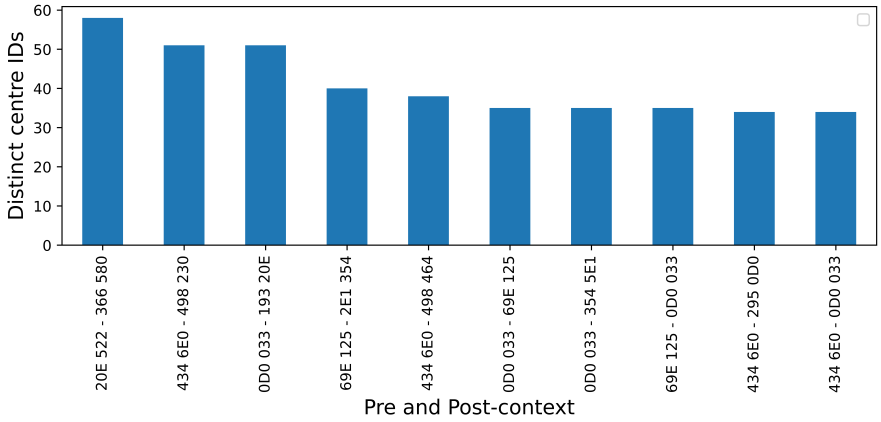
Importance of pre and post-context

We compare the effect of using pre and post-context to predict a CAN ID. Figure 5.7 shows the results for the top 10 CAN ID sequences. Figure 5.7a shows the number of possible distinct CAN IDs given two pre-context IDs as the context. For example, given the context as '69E 125', there are 100 CAN IDs that can appear as the next CAN ID. In contrast, as shown in the Figure 5.7b, given the pre-context of '69E 125' and the post-context of '2E1 354' there are only 40 CAN IDs eligible as the centre ID. Increasing the context size will further reduce this number. This makes the centre ID prediction much more reliable for anomaly detection tasks in CAN ID sequences than the next ID prediction.

After analyzing both benign and attack CAN traffic, our main finding is that most CAN IDs exhibit periodic behaviour that creates a finite set of ID sequences for a fixed window size (e.g., a window of ten consecutive IDs). Attacks on the CAN bus can potentially



(a) The top 10 distinct next ID count for the given pre-context



(b) The top 10 distinct centre ID count for the given pre and post-context

Figure 5.7: Top 10 distinct next and centre ID counts for the given context

disrupt this periodic behaviour, leading to the creation of new sequences. Additionally, injection and suspension attacks change the time between consecutive attack IDs. Carefully trained machine learning algorithms can detect these subtle changes in CAN ID streams, forming the basis for the proposed IDS.

5.4.4 Results and Discussion

The detection accuracy of the GRU model of CAN-CID depends on the centre word prediction accuracy. We expect accurate predictions for benign frames and inaccurate predictions for attack frames to detect weak anomalies. To identify the optimum context from both sides of the centre ID, we experimented with different numbers of IDs as the context, aiming for the highest prediction accuracy using a sample from the benign dataset. Similarly, we explored various numbers of previous IDs for the CAN-NID GRU model. As depicted in Figure 5.8, the CAN-CID model achieved an accuracy of

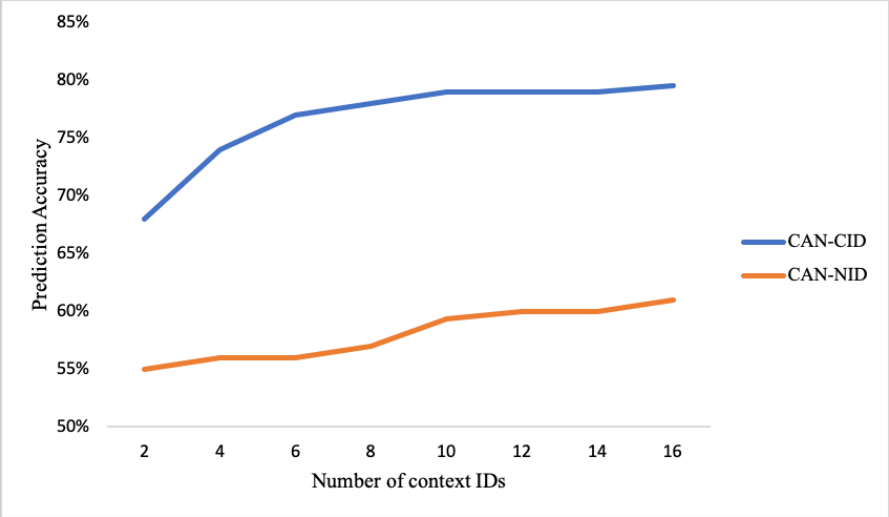


Figure 5.8: Comparison of centre ID (CAN-CID model) and next ID (CAN-NID model) prediction accuracy

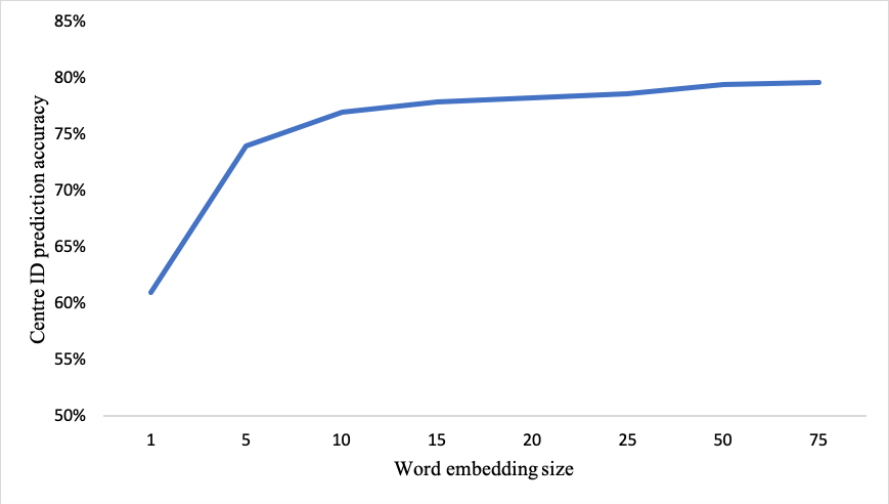


Figure 5.9: Accuracy improvement with word embedding size for the CAN-CID model

80%, while CAN-NID attained a maximum accuracy of 61% with 16 context IDs. This underscores the efficacy of the CBOW approach for CAN sequences. Considering computational efficiency, we chose ten context words (79%) from each side for CAN-CID and 12 context words (60%) for CAN-NID.

The size of word embeddings is a crucial factor affecting both accuracy and computational efficiency. In light of this, we conducted experiments on the CAN-CID model with various

embedding sizes, as illustrated in Figure 5.9. This revealed that accuracy improved up to an embedding size of 50. Consequently, we opted for an embedding size of 50 for both GRU models.

The F1-scores, FPRs, and FNRs of the CAN-CID and CAN-NID models, along with two baseline models, are presented in Table 5.2 and Table 5.4 for the ROAD dataset. Table 5.2 and Table 5.4 report fabrication and masquerade attacks, respectively, where the best performance (F1-score) for each attack is shown in bold. As depicted in the tables, the CAN-CID model outperforms the two baseline models for every attack and achieves a 100% F1-score for six attacks. More importantly, this model achieved 0% or very small FPR and FNR values, which are critical aspects for an IDS. The CAN-NID model also outperformed baseline models for seven attacks. A fuzzing attack is relatively easy to detect due to illegal ID injection, and therefore, all models except the transition model achieved an F1-score of 100%. However, correlated signal and correlated signal masquerade attack detection rates are low compared to other attacks. This may be attributed to them targeting the second most frequent ID, which has a slightly random transmission rate compared to other IDs. As a result, it creates more sequences, leading to the generation of more valid sequences even for attack frames. This is a limitation of the proposed model, whereby it achieves a lower detection rate for attacks that target IDs with random transmission rates. However, a greater number of CAN IDs have fixed transmission rates [9], and therefore, CAN-CID can detect the majority of injection attacks. Furthermore, since CAN IDs have fixed transmission rates, most ID sequences are likely to be independent of driving behaviours. This makes the model resilient to such changes. However, one of the limitations of the proposed model is that the CAN-CID model requires greater variety in the benign data to minimize the occurrence of unseen CAN ID sequences and time intervals.

Figure 5.10 provides a comparison of the attack detection performance between time-based and GRU models. Specifically, Figure 5.10a displays the results for fabrication attacks, while Figure 5.10b showcases masquerade attacks. Typically, the time-based model demonstrates a higher F1-score in detecting fabrication attacks but falls short in detecting masquerade attacks. Conversely, the GRU model exhibits superior performance in detecting both types of attacks, achieving a higher F1-score. However, it is noteworthy that the time-based model outperforms the GRU model for correlated signal and reverse light on attacks. This could be due to the high frequency of both IDs, which likely generates some benign sequences even during the attack duration. However, since injection disrupts the interarrival time, the time-based model can detect these attacks

Table 5.3: Comparison of CAN-CID and CAN-NID models and baseline models detection performance for fabrication attacks (ROAD dataset)

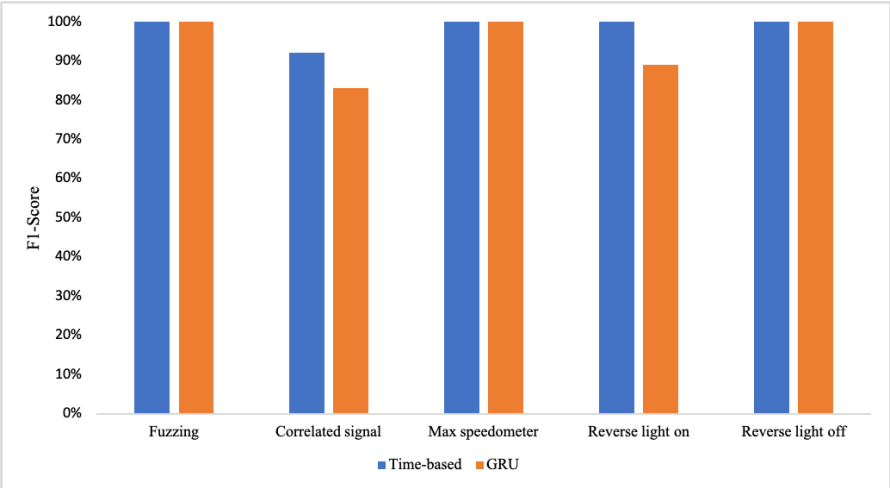
Attack	Model	F1-score	FPR	FNR
Fuzzing	Transition matrix	71%	48%	0%
	N-gram	**100%**	0%	0%
	CAN-NID	**100%**	0%	0%
	CAN-CID	**100%**	0%	0%
Correlated signal	Transition matrix	90%	10%	6%
	N-gram	27%	0%	100%
	CAN-NID	78%	21%	42%
	CAN-CID	**91%**	2%	12%
Max speedometer	Transition matrix	79%	27%	0%
	N-gram	89%	0%	29%
	CAN-NID	94%	1%	2%
	CAN-CID	**100%**	0%	0%
Reverse light on	Transition matrix	63%	57%	0%
	N-gram	87%	0%	29%
	CAN-NID	94%	1%	2%
	CAN-CID	**100%**	0%	0%
Reverse light off	Transition matrix	92%	9%	0%
	N-gram	94%	0%	16%
	CAN-NID	**100%**	0%	17%
	CAN-CID	**100%**	0%	0%

Table 5.4: Comparison of CAN-CID and CAN-NID models and baseline models detection performance for masquerade attacks (ROAD dataset)

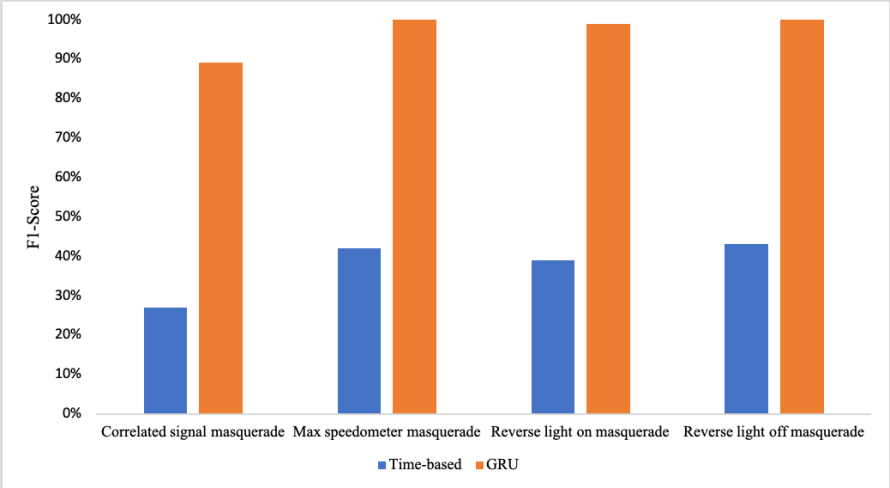
Attack	Model	F1-score	FPR	FNR
Correlated signal masquerade	Transition matrix	38%	10%	86%
	N-gram	27%	0%	100%
	CAN-NID	64%	22%	57%
	CAN-CID	**89%**	4%	10%
Max speedometer masquerade	Transition matrix	79%	27%	0%
	N-gram	99%	0%	1%
	CAN-NID	86%	0%	36%
	CAN-CID	**100%**	0%	0%
Reverse light on masquerade	Transition matrix	63%	57%	0%
	N-gram	87%	0%	29%
	CAN-NID	94%	1%	2%
	CAN-CID	**99%**	0%	1%
Reverse light off masquerade	Transition matrix	92%	9%	0%
	N-gram	94%	0%	16%
	CAN-NID	**100%**	0%	7%
	CAN-CID	**100%**	0%	0%

with a higher detection rate. Training with a substantially large dataset that includes various benign driving behaviours might improve the GRU model's performance on these attacks, as it will reduce the probability of these sequences compared to other benign sequences. The ensemble IDS achieves the performance of the best individual model for all attacks.

As previously mentioned, we utilised two HCRL datasets to assess the generalization capability of the proposed model. The outcomes obtained from these datasets closely



(a) Time-based and GRU model detection performance for fabrication attacks



(b) Time-based and GRU model detection performance for masquerade attacks

Figure 5.10: Time-based and GRU model detection performance comparison

mirror the results observed with the ROAD dataset (refer to Table 5.5 and Table 5.6). Notably, both the CAN-CID and CAN-NID models exhibited superior performance compared to the baseline models. However, it is important to highlight that the baseline models performed relatively better on the HCRL datasets than on the ROAD dataset. This disparity might be attributed to the HCRL datasets featuring a limited number of IDs, resulting in fewer CAN ID sequences generated compared to the ROAD dataset. This limitation potentially contributes to a higher level of predictability in the HCRL datasets.

Table 5.5: Comparison of attack detection performance of the CAN-CID and CAN-NID models and the baseline models for the HCRL CH dataset

Attack	Model	F1-score	FPR	FNR
DoS	Transition matrix	75%	52%	0%
	N-gram	96%	10%	0%
	CAN-NID	97%	6%	0%
	CAN-CID	**99**%	1%	0%
Fuzzy	Transition matrix	91%	20%	0%
	N-gram	94%	14%	0%
	CAN-NID	97%	6%	0%
	CAN-CID	**100%**	0%	0%
Gear Spoofing	Transition matrix	98%	4%	0%
	N-gram	98%	4%	29%
	CAN-NID	99%	1%	1%
	CAN-CID	**100%**	0%	0%
RPM Spoofing	Transition matrix	86%	28%	0%
	N-gram	98%	4%	0%
	CAN-NID	**99%**	0%	2%
	CAN-CID	**99%**	0%	2%

Table 5.6: Comparison of attack detection performance of the CAN-CID and CAN-NID models and the baseline models for the HCRL SA dataset

Attack	Model	F1-score	FPR	FNR
Flooding	Transition matrix	89%	28%	0%
	N-gram	99%	2%	0%
	CAN-NID	**100%**	0%	0%
	CAN-CID	**100%**	0%	0%
Fuzzy	Transition matrix	85%	28%	0%
	N-gram	99%	1%	0%
	CAN-NID	99%	1%	0%
	CAN-CID	**100%**	0%	0%
Malfunction	Transition matrix	68%	54%	0%
	N-gram	84%	28%	0%
	CAN-NID	91%	2%	17%
	CAN-CID	**96%**	0%	4%

Detection latency is another criterion that we focused on improving as it is vital for moving vehicles. Table 5.7 compares the average detection latency for CAN-CID, CAN-NID and the two baseline models. The IDS monitors CAN traffic for 100ms and gives the prediction in 10ms. CAN-CID outperformed CAN-NID and the two baseline models. The small amount of time required for monitoring and prediction allows the vehicle driver or the vehicle itself to take appropriate countermeasures. Consequently, considering both detection capability and latency, the proposed algorithm emerges as a practically deployable solution for detecting cyberattacks on the CAN bus. Moreover, relying solely on CAN ID and time as features for the ensemble model contributes to improved detection latency in resource-constrained environments. Additionally, this model is likely to exhibit superior generalization capability compared to payload-based models, as data specifications (payload) may undergo significant changes across various vehicle makes

Table 5.7: Average detection latency comparison for a 100 ms prediction window

Model	Detection latency (ms)
Transition matrix	36
N-gram	452
CAN-NID	12
CAN-CID	10

and models.

5.5 Conclusion

Drawing inspiration from prior work on CAN ID-based IDS and Word2Vec technique in NLP domain, this chapter introduced CAN-CID, a novel context-aware ensemble IDS designed to enhance CAN bus security. CAN-CID combines a GRU network with a time-based model. The GRU-based model focuses on predicting the centre ID within a CAN ID sequence, utilising ID-based probabilistic thresholds to identify anomalous IDs. Meanwhile, the time-based model employs time-based thresholds to pinpoint anomalous IDs. The classification of window status as anomalous or benign is determined by assessing the ratio of anomalies to the total number of IDs within an observation window. The performance evaluation of the proposed model is conducted using three publicly available CAN attack datasets.

Our experiments revealed that CAN-CID significantly enhanced overall attack detection performance, surpassing the performance of two baselines and a variant of the proposed model. Prior CAN ID-based IDSs in the literature demonstrated proficiency in detecting injection attacks but fell short in detecting masquerade attacks, as the latter typically do not significantly alter the frequency or sequential behaviour of CAN data. In contrast, the GRU-based model in CAN-CID demonstrates the capability to detect masquerade attacks, particularly when the attacker fails to synchronize with the legitimate ECU's timing. This is attributed to even a slight time shift or a lack of messages from a particular ECU for a brief period during the masquerade attack, which has the potential to generate new CAN ID sequences. Furthermore, the proposed CAN-CID model exhibits low detection latency, a crucial characteristic for a deployable in-vehicle IDS. The ensemble approach employed in CAN-CID enhances attack detection capabilities by leveraging the strengths of independent GRU and time-based models. Given the complexity of CAN data and the diverse characteristics of potential attacks, this study underscores the necessity of an ensemble model with optimized components for different fields of CAN data.

Despite the promising results achieved by the GRU-based model, it inherits a fundamental limitation due to the use of only benign data for model training. The diversity of benign behaviours exhibited by vehicles necessitates a substantial training dataset that encompasses a wide range of driving scenarios. This is similar to the requirement for extensive datasets in training Large Language Models (LLMs), which are trained to predict the next word during model training. This limitation is applicable to any one-class-based IDS proposed based on the CAN ID data. Therefore, addressing this constraint is important to further enhance CAN ID-based IDSs for real-world deployment. Consequently, the next chapter will address this limitation.

Chapter 6

On-device Streaming Learning to Improve CAN ID-based IDS

The CAN ID-based IDS proposed in Section 3.3.1 demonstrates its effectiveness against common cyberattacks on the CAN Bus. While the time-based model exhibits promising results, the GRU-based model demonstrates comparative outcomes for injection attacks and significantly improved results for masquerade attacks involving slight time shifts with the legitimate ECU. However, as emphasized in the preceding chapter, CAN-ID-based IDS, which relies on one-class data for training, necessitates a substantial amount of data to accurately profile normal behaviour. To address this limitation, this chapter introduces a novel approach to retrain the IDS using streaming CAN data. Furthermore, it proposes an effective technique to prevent data poisoning attacks during IDS retraining. This chapter addresses the RQ2.

The main findings of this chapter were published at the Symposium on Vehicle Security and Privacy (VehicleSec) in the Network and Distributed System Security (NDSS) Symposium 2023 [191] and have been submitted to 37th IEEE Computer Security Foundations Symposium 2024.

6.1 Introduction

One-class classification-based IDSs have proven successful in IVNs, demonstrating the ability to detect a wide range of attacks while relying solely on benign data for training the algorithms [26, 58]. However, a significant drawback of this approach is the necessity

for a large, representative sample of benign data that accurately reflects various driving scenarios. A small training dataset increases the risk of higher false positives or negatives. Collecting an extensive and diverse dataset that adequately represents benign driving behaviours poses challenges, and training a deep learning model with such a dataset is computationally expensive.

An effective approach to address this challenge involves implementing streaming learning techniques. This strategy has been successfully applied in domains like the Internet of Things (IoT) for continual improvement of IDSs over time [192, 193, 194]. As a potential avenue for future research, this approach was initially proposed in the context of in-vehicle IDS in [195], suggesting incremental training of the model with real-time data. However, the implementation of AI-based IDS retraining with streaming data faces challenges in in-vehicle network environments due to computational and storage limitations [116]. Only one previous work adopted streaming learning to train the IDS on the CAN bus [196]. In this work, they employed the streaming data Isolation Forest (iForestASD) algorithm, utilising message timing information as the sole feature. However, the model did not yield promising results, primarily due to its reliance solely on time as a feature. To tackle the challenges mentioned earlier, this chapter introduces an on-device transfer learning technique (CAN-ODTL) aimed at retraining the classification layer of the GRU-based model proposed in Chapter 5 on a resource-constrained Raspberry Pi device.

Despite the advantages provided by AI models, they are vulnerable to adversarial attacks such as model poisoning and data poisoning [197]. Of these attacks, streaming learning is particularly vulnerable to data poisoning [187]. Detecting data poisoning attacks during in-vehicle intrusion detection training with streaming data is more difficult than in non-streaming training due to the high CAN bus data transmission rate and limited computational resources. Therefore, to overcome these challenges, this chapter proposes an effective defence technique against data poisoning attacks during the CAN-ODTL procedure.

6.2 Chapter Contribution

This chapter introduces an on-device transfer learning technique (CAN-ODTL) designed to retrain the classification layer of the GRU-based IDS using streaming CAN data. Additionally, it presents a defense mechanism against data poisoning attacks targeting the CAN-ODTL procedure. The primary contributions of this chapter can be summarized as follows:

1. Proposes an on-device transfer learning technique, named as CAN-ODTL, to retrain the CAN GRU-based IDS in a resource-constrained environment. The proposed model addresses the need for having a large benign dataset to train the algorithm by incremental retraining the classification layer on the IDS with streaming CAN data.

2. The CAN-ODTL employs an optimized data pre-processing algorithm to enhance the pre-processing of streaming CAN data, contributing to improved detection latency while minimizing CPU and RAM usage on the Raspberry Pi. Additionally, the quantization technique is utilised in CAN-ODTL to reduce the model size and enhance detection latency.

3. This chapter emphasizes the impact of data poisoning attacks on the retraining process of the CAN-ODTL procedure, showcasing their capability to substantially diminish the IDS's performance. In response to this challenge, this introduces a novel approach involving hierarchical clustering to identify CAN bus ID clusters. By leveraging these clusters, the proposed technique can effectively detect anomalous behaviour in CAN data.

4. Introduce a defence technique specifically designed to counter data poisoning attacks aimed at CAN-ODTL retraining with streaming data. The proposed algorithm effectively identifies poisoned samples before the retraining process and removes them from the dataset.

6.3 CAN IDS On-Device Transfer Learning (CAN-ODTL)

As mentioned earlier, training one-class classification-based CAN IDSs necessitates a substantial dataset that accurately captures diverse benign driving behaviours. However, acquiring datasets that encompass all types of benign driving behaviours poses practical challenges. A more appropriate way to address this issue involves on-device training in an incremental manner until the model is trained over an extended duration with more data. This can be accomplished by deploying the algorithm on the CAN bus and training the algorithm while it is driving. Nevertheless, due to the limited computational power of ECUs, they are incapable of training computationally expensive deep learning or shallow learning models. Since it is possible to access the CAN bus data through the OBD-II port or CAN gateway, the algorithm can be deployed in a small device like Raspberry Pi and connected to the CAN bus. Raspberry Pi, despite being a versatile embedded computing

device, comes with constrained memory and processing power. Even though it is used for machine learning and deep learning inferences, model retraining is challenging due to the limited computational resources [198]. To overcome this, we propose a transfer learning technique to incrementally retrain the classification layer of the deployed model. This technique involves utilising transfer learning to update the classification layer of the pre-trained model using additional data, thereby enhancing the intrusion detection rate. This process is depicted in Figure 6.1. Given that most CAN IDs have a predefined transmit rate, we assume this behaviour remains constant over time, indicating no concept drift or data drift associated with CAN ID transmission. Thus, this model focuses on learning the majority of benign sequences through incremental learning, aiming to minimize the occurrence of unseen benign sequences and enhance attack detection. However, if any data drift occurs due to wear and tear or other reasons, the model can be retrained at regular intervals to adapt to these changes.

The CAN-ODTL utilise the GRU-based model proposed in Section 3.3.1 with an additional self-attention layer. This allows capturing the important information in CAN sequences for long CAN ID sequences. The model comprises two parts: an encoder, denoted as f_ϕ, and a decoder, denoted as g_θ. The encoder includes the first three layers of the model, and the output of the attention layer serves as the input to the decoder, which functions as the classification layer. The complete model is first trained with a sufficiently large benign dataset, aiming to minimize the objective function of categorical cross-entropy, as given in Equation 5.3. Following pre-training, the encoder model is converted into a more lightweight TFLite version. This conversion also involves applying quantization to reduce the encoder's detection latency. The on-device transfer learning retraining procedure is outlined in Algorithm 2. In Raspberry Pi experiments, CAN data logs were saved on the micro SD card and read from there.

The algorithm takes the pre-trained encoder, streaming CAN data, a data buffer for storing the streaming data, the sequence window size, and batch size as inputs. First, it preprocesses the data, extracting numerical CAN IDs from the streaming CAN frames. A Python double-ended queue (deque) is employed during data preprocessing for its efficient support of data append and pop operations from both ends with O(1) time complexity. Extracted IDs are stored in the buffer since streaming CAN data speed surpasses the retraining process speed. However, this storage can also be done on the SD card, as it does not necessitate inferencing during the retraining period. A batch of context IDs from the buffer is input to the pre-trained encoder. The output of the encoder along with the batch of centre IDs are used to retrain the classification layer

using transfer learning. This is initialized from the pre-trained weights W_4. Each batch is trained for one epoch. A learning rate decay schedule is employed to enhance the incremental training process. During the retraining, updated W_4 weights are maintained in memory until the retraining cycle concludes, which, in our experiments, marks the end of a CAN log. In real deployments, saved CAN logs are not available, requiring the use of streaming CAN data. Retraining can begin at the start of each driving session and end at the session's conclusion. This process should continue until the system learns the majority of CAN IDs, with the duration depending on the number of CAN IDs in the vehicle and their transmission behaviour. After triggering the retraining completion, learned weights are saved to the SD card for inferencing or future retraining. Once the model is trained to a satisfactory level such as a few weeks or months of training, the trained decoder is converted into the TFLite with the quantization for the inferences. We retrained the model for the available CAN logs. Using the learned weights, this model can be retrained again as and when necessary or based on a pre-defined schedule. After executing the retraining procedure for each CAN log, we evaluated the model's progress by monitoring the centre ID prediction accuracy. This was done using a small sample of the benign dataset that had been saved to the SD card.

CAN-ODTL anomaly detection procedure is similar to the GRU-based model anomaly detection procedure outlined in Algorithm 1. However, it exclusively focuses on the GRU model without incorporating the time-based model. Furthermore, it employs the quantized TFLite model, which is the output of Algorithm 2, $g_\theta(f_\phi(x))$, as the trained model to achieve lower detection latency. The steps of the CAN-ODTL anomaly detection procedure are shown in Algorithm 3.

6.4 Evaluation and Performance Results - CAN-ODTL

This section discusses the parameters of the algorithm and the performance evaluation for the CAN-ODTL technique. The ROAD dataset [30] is utilised for experiments, incorporating fuzzing, max speedometer, reverse light on and off, along with their masquerade attacks.

6.4.1 Experimental Setup

For the initial model training, different subsets of benign data are used by merging into one dataset without changing the temporal behaviour of the CAN data. A separate benign sample is used as the testing set to evaluate the model loss and accuracy. Similarly,

Algorithm 2 CAN-ODTL retraining procedure

Input: Pre-trained encoder f_ϕ, Streaming CAN data F, Buffer size Z, Window size W, Batch size B
Output: Retrained model $g_\theta(f_\phi(x))$
  Init: $ID\_list = [\,] \in Z$
  while F is not empty **do**
    Read l in F ▷ Read line by line in F
    Pre-process l
    Extract id
    Append id to ID_list ▷ Parallel append
    while len$(ID_list) \geq W$ **do**
      Init: Empty arrays X,Y
      while len$(X) \leq B$ **do**
        Append earliest context ids in W to X
        Append earliest center id in W to Y
        Remove earliest id from ID_list
      end while
      $z = f_\phi(X)$ ▷ Output of the attention layer
      $\hat{Y} = g_\theta(z)$ ▷ Retraining
    end while
    Save W_4
    Convert g_θ to TFLite format
  end while

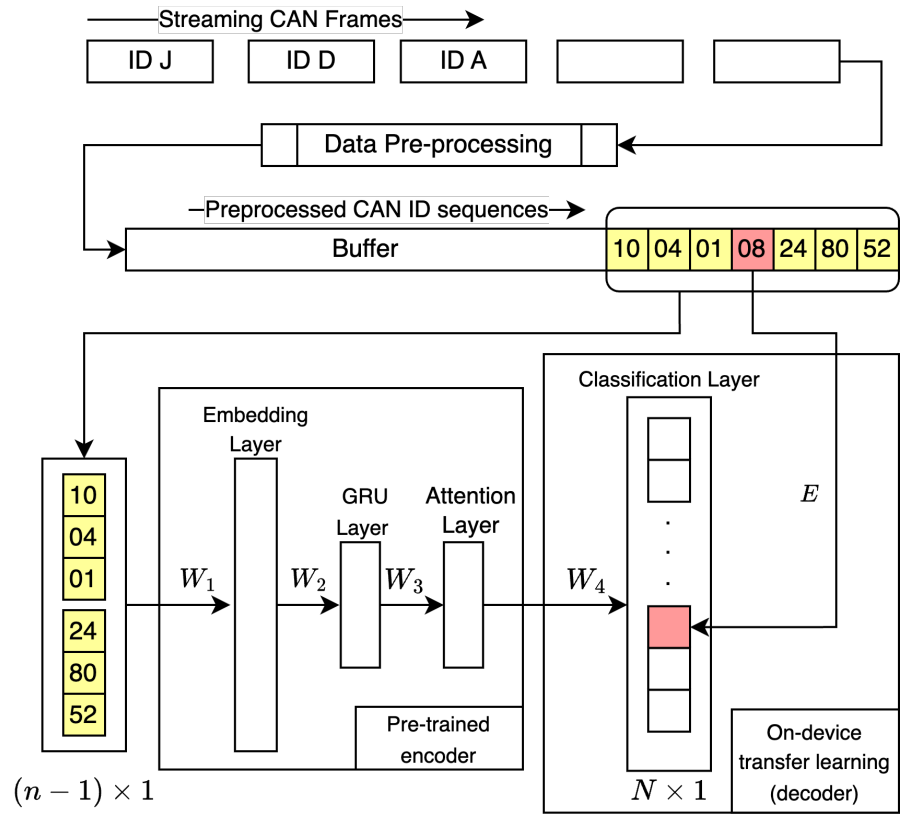


Figure 6.1: On-device transfer learning retraining procedure

a representative dataset sample is employed for threshold estimation. Hyperparameters are optimized through grid search. Given that the model's anomaly detection capability depends on the accuracy of centre ID predictions, it is anticipated to achieve high softmax probabilities for benign centre IDs and low probabilities for anomalous centre IDs. Accuracy is computed based on the true and predicted IDs with the highest softmax probability, serving as a metric to monitor model progress. This is used to monitor the model progress. Both model accuracy and the number of training parameters are considered in hyperparameter selection to obtain a lightweight model. Consequently, a window size of 11 is chosen, with 5 IDs each assigned as pre and post-context from each side of the centre ID. The rationale behind this choice is elaborated in Section 5.4.2. The embedding size is set to 50 and only 16 nodes are used in the GRU Layer. The classification layer includes 107 classes, in which one class is allocated for new IDs, which can appear due to some injection attacks such as fuzzing attacks. These numbers are equivalent to the CAN GRU model proposed in Chapter 5, with the exception of the 16 nodes in the GRU layer, a modification introduced by the incorporation of the self-attention layer.

Algorithm 3 CAN-ODTL anomaly detection

Input: Streaming CAN data F, Anomaly threshold ω, Window threshold ψ, Time window T, Trained model $g_\theta(f_\phi(x))$
Output: Anomaly status for each window

while F is not empty **do**
  read x, y, time_stamp t, t_min
  while $t - t\_min \leq T$ **do**
    Init: Benign count $C_b = 0$, Anomaly count $C_a = 0$
    $\hat{y} = g_\theta(f_\phi(x))$ ▷ Quantized TFLite model
    if $\hat{y} < \omega$ **then** ▷ for id y
      Declare x as a weak anomaly
      $C_a = C_a + 1$
    else
      Declare x as a benign
      $C_b = C_b + 1$
    end if
  end while
  if $C_a/(C_a + C_b) > \psi$ **then**
    Return Anomaly
  else
    Return Benign
  end if
  $t\_min \leftarrow t$
end while

Due the usage of a self-attention layer, a comparable level of accuracy is achieved with the 16 GRU nodes. The initial model is trained to 100 epochs with 128 batch size. Early stopping is used to avoid overfitting. Stochastic Gradient Descent (SGD) is chosen as the optimizer, with a learning rate of 0.01 due to its strong generalization capabilities for model retraining [199]. During the classification layer retraining, the learning rate is decreased by a factor of 1.1 every 5000 epochs to avoid overfitting. These parameter configurations are selected through iterative experiments to achieve optimal retraining for accuracy and efficiency.

Given the frame transmission rate of approximately 2500 frames per second in the ROAD dataset, the attack datasets are divided into 100 millisecond windows for the identification of attack windows. The window threshold is set to 0.01 based on the lowest false positive rate (average) for the benign dataset. For calculating the anomaly threshold for each CAN ID, a small margin is allowed for unseen benign data, and 0.001^{th} quantile values are considered. Tensorflow and Keras libraries are used with Python 3.8 for the

implementation, whereas a custom training loop is used during the retraining. KerasTuner is used for the grid search. All pre-training experiments run on a MacBook M1 Pro with 16 GB RAM. For on-device transfer learning, a Raspberry Pi 4 Model B (8GB version) is used, equipped with a 16GB micro SD card. TensorFlow 2.10 and Python 2.8 are installed on the Raspberry Pi for model retraining, with TFLite runtime being used for inference.

6.4.2 Requirement for Streaming Learning

Figure 6.2 illustrates the count of unique contexts of size two for a subset of CAN IDs during a 30-minute driving dataset from the ROAD dataset. For this analysis, we selected one ID as the pre-context and another ID as the post-context. The majority of the 106 CAN IDs in the ROAD dataset have a large number of unique contexts. For instance, CAN ID 6E0 has over 800 unique contexts. Consequently, when aiming to predict the centre ID, such as 6E0, with a higher softmax probability, training on an extensive set of sequences becomes imperative. The CAN bus transmits high, medium, and low-frequent IDs, leading to an imbalanced ID distribution. Figure 2.6 in Section 3.3.2, illustrates the number of frames transmitted for selected IDs within a one-second period. This includes high, medium and low frequent IDs. Specifically, a large number of high-frequent CAN IDs coexist with a small number of medium and low-frequent IDs within a specific time range. This also necessitates a comprehensive dataset to enhance the prediction capability for medium and low-frequency IDs and mitigate prediction bias towards high-frequency IDs. The ID-based IDSs proposed in the literature [94, 200, 85, 64, 58] utilised ID sequences for attack detection. Consequently, all of these models should train with diverse and extensive benign datasets to differentiate anomalous sequences effectively. Therefore, the adoption of a streaming learning approach is more suitable for training ID-based CAN IDS. This approach offers the advantage of improving attack detection while utilising less computing resources.

Effect of dataset size

We experimented with different dataset sizes as the initial training dataset to identify the effect of the dataset size and centre word prediction accuracy to monitor the model performance. Figure 6.3 shows the accuracy for the evaluation dataset and overall F1-score for the selected attack datasets. The accuracy for the evaluation dataset is calculated by considering the ID that obtained the highest softmax probability. Since multiple attacks, including masquerade attacks, are taken into account, the overall F1-score is used for a

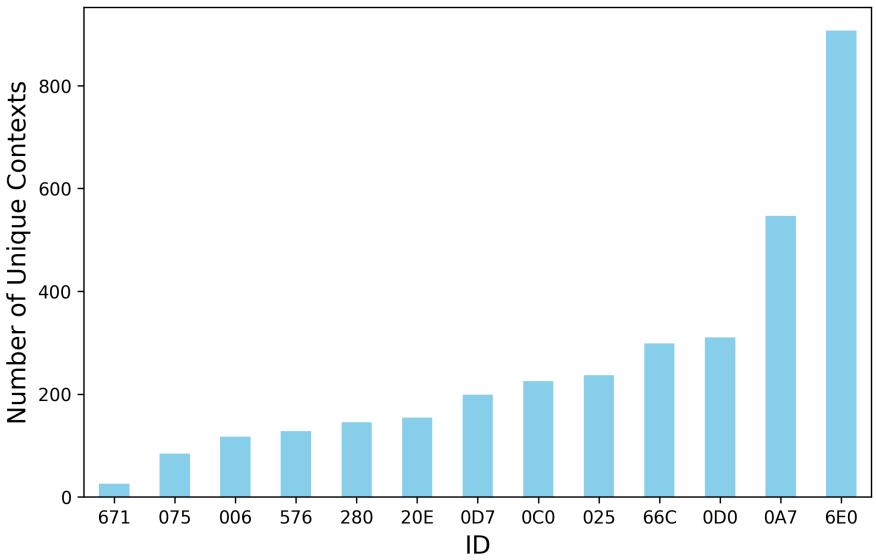


Figure 6.2: Number of unique contexts of window size two for a subset of CAN IDs

fair comparison. The results indicate that centre ID prediction can be used to monitor the model progress and attack detection capability is improved with the dataset size.

All layers retraining and last layer retraining

The retraining capability of the model relies on the learning acquired during pre-training. Therefore, to evaluate the effect of the pre-trained model on the retraining process, we used four pre-trained models trained on four dataset sizes. Then we retrained the models using an additional benign dataset (2000000 frames) as full model retraining and last layer retraining. The results are illustrated in Figure 6.4. The findings indicate that achieving good accuracy through retraining necessitates a pre-trained model trained on a sufficiently large dataset. This requirement may arise because, during retraining, only a single epoch is considered in comparison to the large number of epochs used during pre-training. With this large initial dataset size, both full model retraining and last layer retraining converge to approximately the same level of accuracy. Therefore, transfer learning proves effective in retraining the last layer with lower computational overhead to achieve the same accuracy level provided there is a good pre-trained model. Focusing on retraining the classification layer alone mitigates the risk of overfitting. A sample of 3000000 CAN frames in ROAD dataset is approximately equivalent to 30 minutes drive dataset. This sample included three benign datasets which covered different benign driving activities such as drive, brake, accelerate and decelerate. Thus, the results suggest

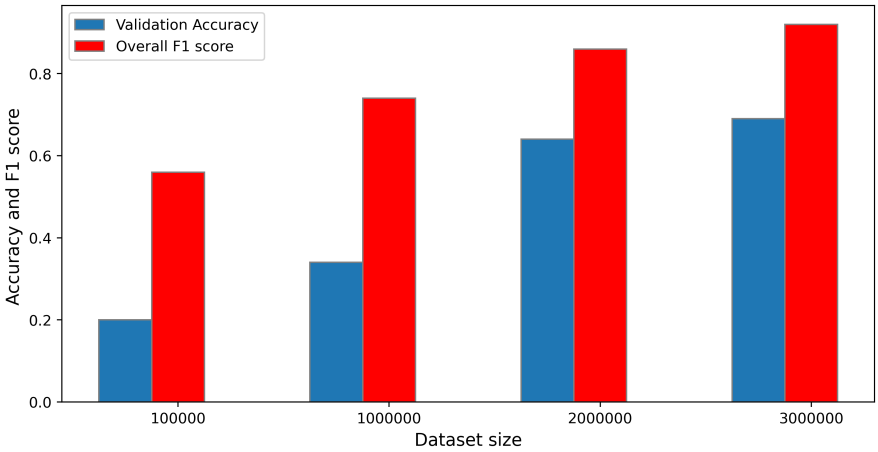


Figure 6.3: Accuracy and overall F1-score improvement with dataset size

the potential to enhance the pre-trained model through transfer learning using a 30-minute to one-hour drive dataset for initial model training. However, this may depend on the number of CAN IDs on the vehicle and the frequency of CAN frame transmissions on the CAN bus. Additionally, incorporating diverse benign driving activities into this dataset is crucial, and ultimately, on-device transfer learning could further improve the model with a large set of streaming data.

In light of these findings, our initial model is trained with 3000000 CAN frames. Subsequently, a streaming dataset of 2000000 frames is employed to retrain the last layer, following the algorithm outlined in Algorithm 3, using Raspberry Pi. We conducted a comparative analysis of CAN-ODTL detection with the base CAN GRU-based model, which utilised 32 nodes without the self-attention layer. Consequently, the base CAN GRU-based model includes an additional 6486 model parameters compared to CAN-ODTL. Furthermore, we assess the performance against our pre-trained model (CAN-PreODTL). Table 6.1 presents the detection performance of CAN-ODTL in comparison to CAN-PreODTL and the base CAN GRU-based model. To evaluate detection performance, macro-averaged F1-score (F1), false-positive rate (FP), and false-negative rate (FN) are used. For CAN-ODTL evaluation, experiments are conducted on a MacBook M1 Pro using the retrained model from Raspberry Pi.

According to this, both base CAN-GRU and CAN-ODTL models achieved a higher detection rate for all the attacks. CAN-ODTL marginally outperforms CAN-CID for reverse light on and reverse light on masquerade attacks. As expected CAN-PreODTL

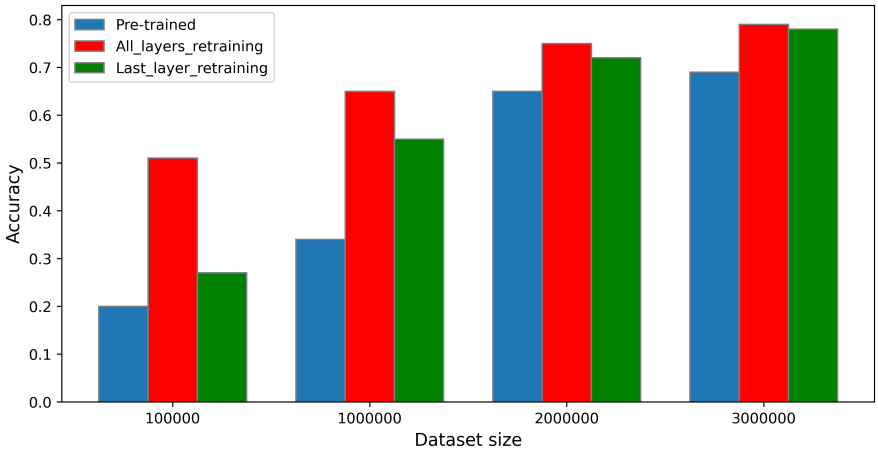


Figure 6.4: Effect of pre-trained model on retraining process

suffers from high false positives. This is due to the unseen benign sequences during the model training. On-device transfer learning helps to reduce false positives as it trains for large dataset. These results indicate the effectiveness of on-device transfer learning to retrain the CAN IDS.

Overhead analysis

Detection latency and memory efficiency are pivotal considerations for a CAN IDS. We conducted a comparison of the mean inference time between CAN-ODTL and the base CAN-GRU model on a MacBook M1 Pro. The base CAN-GRU model exhibits an inference time of 0.1ms, while CAN-ODTL achieves a faster 0.08ms. This enhancement in inference time is attributed to the reduced number of model parameters in CAN-ODTL compared to the base CAN-GRU model. Consequently, CAN-ODTL achieves an equivalent or marginally improved detection level with a lower detection latency. During the transfer learning on Raspberry Pi, we monitor resource consumption. On average, it takes 0.16s to retrain the classification layer for one batch, consisting of 256 sequence windows. This implies approximately 0.625ms to retrain one input window. The retraining process can handle around 1600 CAN frames per second, while the CAN bus transmits around 2500 frames per second. This demands having a buffer with at least 900 CAN frames. Generally, it takes around 75KB of memory to keep 900 CAN frames. Given the 8GB of available memory, this can be efficiently managed. During the model retraining, on average, it utilised 45% CPU and 157MB of RAM. For inference, the Tensorflow model is converted into the TFLite version, offering three options: non-optimized

Table 6.1: Comparison of CAN-ODTL, CAN-PreODTL and baseline model detection performance of ROAD dataset

Attack	Model	F1	FP	FN
Fuzzing	CAN-GRU	**100**%	0%	0%
	CAN-PreODTL	79.4%	18.5%	0%
	CAN-ODTL	**100**%	0%	0%
Max speedometer	CAN-GRU	**100**%	0%	0%
	CAN-PreODTL	83.1%	21.3%	0%
	CAN-ODTL	**100**%	0%	0%
Max speedometer masquerade	CAN-GRU	**100**%	0%	0%
	CAN-PreODTL	83.1%	21.3%	0%
	CAN-ODTL	**100**%	0%	0%
Reverse light on	CAN-GRU	99.4%	0%	0.3%
	CAN-PreODTL	95.6%	6.9%	0%
	CAN-ODTL	**99.8**%	0.1%	0%
Reverse light on masquerade	CAN-GRU	99.1%	0%	1.0%
	CAN-PreODTL	96.2%	6.6%	0%
	CAN-ODTL	**99.9**%	0%	0%
Reverse light off	CAN-GRU	**100**%	0%	0%
	CAN-PreODTL	85.4%	17.0%	0%
	CAN-ODTL	**100**%	0%	0%
Reverse light off masquerade	CAN-GRU	**100**%	0%	0%
	CAN-PreODTL	85.4%	17.0%	0%
	CAN-ODTL	**100**%	0%	0%

default quantization, dynamic range quantization, and float 16 quantization.

Model sizes and average inference times are shown in Table 6.2. The conversion from Tensorflow to TFLite significantly reduces the model size and inference time on Raspberry Pi. Both the default TFLite quantization model and the float 16 quantization model produce softmax probabilities comparable to the Tensorflow model outputs, matching up to the fifth decimal point. Consequently, these two models achieve the same detection level as the Tensorflow model, as illustrated in Table 6.1. However, the dynamic range quantization model exhibits a slight drop in accuracy, despite having the smallest model size. Considering the model size and inference time on Raspberry Pi, the float 16 quantization model outperforms other variations, demonstrating higher intrusion detection capability. In comparison to the Tensorflow model, this translates to a 78% reduction in model size and an 83% improvement in inference time. During the inference time, it utilises an average of 38% CPU and 5MB of RAM on Raspberry Pi. It takes only 0.5ms for a one-frame prediction, including data pre-processing. This is approximately in line with the CAN data transmission rate of the ROAD dataset. For a 100ms observation window, on average, it takes 125ms to provide the prediction. Generally, average driver response time ranges from 0.7s to 1.5s [201]. Therefore, the 125ms detection time is minimal compared to the driver's response time. These experimental results on Raspberry Pi underscore the effectiveness of the on-device transfer learning process and the inference of the proposed

Table 6.2: CAN-ODTL model inference overhead on Raspberry Pi

Model	Model size (KB)	Inference time (ms)
Tensorflow	233	3
Default TFLite quantization	82	0.7
Dynamic range quantization	29	0.5
Float 16 quantization	49	0.5

method for near real-time intrusion detection in the CAN bus.

6.5 Preventing Data Poisoning Attacks During CAN-ODTL with Streaming Data

Despite the promising results achieved with CAN-ODTL for IDS retraining with streaming data, there is a potential risk of data poisoning attacks that could compromise the model's performance. Therefore, this section proposes an effective defence technique against data poisoning attacks targeting CAN-ODTL with streaming data.

6.5.1 Threat Model

This section elaborates on the threat model employed in the data poisoning experiments.

Label Flipping Attack

The CAN-ODTL based IDS employs CAN ID sequences as its input data, where these sequences are represented as categorical labels. For example, if the chosen window size is 5, an ID sequence may look like 'ID-A', 'ID-B', 'ID-C', 'ID-D', 'ID-E'. The objective of the model is to predict the centre ID ('ID-C') given the pre and post-context. Thus, the input data would be 'ID-A', 'ID-B', 'ID-D', 'ID-E', with the label (target) being the centre ID, which in this case is 'ID-C'. As this approach utilises sliding windows, for the subsequent window, 'ID-C' becomes part of the input data, while 'ID-D' becomes the new label. Consequently, changing the label of the input data can be viewed as dirty-label poisoning whereas label-flipping in this case, modifies both the label and content of the training data.

Since the input data does not incorporate any other features, the most realistic way to poison the data is through CAN ID injection, which is similar to CAN injection attacks or stop transmitting one or more CAN IDs, which is similar to CAN suspension attacks. Both of these data poisonings create anomalous CAN ID sequences which cannot be observed during benign driving. These attacks can be either targeted or untargeted

forms of poisoning. Given the unavailability of realistic poisoning data and the similarity between data poisoning with the injection and suspension attacks in this context, we generate synthetic poisoned data samples for model retraining. To achieve this, we first analyze the CAN ID changing behaviour during realistic injection attacks and replicate it when creating the synthetic poisoned data. One-hour benign driving dataset is selected and divided into 60 subsets, each spanning one minute while maintaining the sequential behaviour of the dataset. We consider a scenario in which a $P\%$ of malicious frames within each one-minute subset are randomly injected by an adversary. This is similar to the approach used in federated learning data poisoning attacks to poison the clients [202]. However, in this case, we poison one-minute subsets of the streaming data as the streaming training setting differs from the federated learning. Considering the available limited benign datasets, one-minute samples are selected to observe the effect of poisoning attacks for 60 benign samples. To replicate the data poisoning similar to a suspension attack, legitimate CAN IDs are removed from the CAN trace for the selected time period t.

Adversary's Goal

The goal of the adversary is to manipulate the learned parameters of the classification layer in the CAN-ODTL based IDS to reduce its ability to correctly identify malicious frames during real CAN bus attacks. This can be achieved by training the model on malicious frames, which are assumed as benign, leading to the model incorrectly classifying them as benign frames during inference.

Adversary's Knowledge and Capacity

In our assumption, the adversary possesses grey-box knowledge about the victim CAN-ODTL model, allowing the adversary to access certain information about the CAN-ODTL and have access to the CAN bus of the vehicle for data injection. The adversary is capable of collecting CAN data of a similar vehicle and using that knowledge to perform data injection and label manipulation, which can poison the training data. However, the adversary lacks the capability to engage in model tampering, which involves manipulating the model's weights and biases.

6.5.2 Defence Against Data Poisoning Attack

Defending against data poisoning attacks that introduce new CAN IDs is relatively straightforward since these attacks involve injecting random CAN IDs into the CAN

Bus. Given that the number of CAN IDs is fixed, it is possible to filter out and discard any unknown CAN IDs from the streaming CAN data prior to retraining.

The data injection attack leads to an increase in the count of targeted IDs during the injection period. By mimicking the data injection attack as a data poisoning attack, the number of CAN IDs associated with the targeted IDs also increases. The extent of message increment depends on the poisoning percentage P within a time window t. Conversely, a suspension attack leads to a reduction in the count of CAN IDs, with the extent of this decrease influenced by the suspension time. Experiments of a wide variety of realistic injection attacks show that these injection attacks only increase the targeted IDs count. Therefore, we leverage this characteristic to detect data poisoning attacks. However, it is important to note that the maximum number of CAN IDs for a specific ID within a fixed window t can vary based on different conditions. Consequently, it is not feasible to determine a fixed threshold for each CAN ID and utilise it to identify anomalies. Analysis of the number of messages within fixed time windows shows that certain sets of IDs maintain relatively fixed ratios. To exploit these associations, we employ hierarchical clustering. Since the number of clusters is unknown in advance, agglomerative hierarchical clustering is used to identify ID clusters. The ward method is selected as the linkage method as it is less sensitive to outliers. Once the clusters are identified, each sample is divided by the minimum ID count within the respective cluster. This allows us to determine the ID ratios within the clusters.

The ID counts for a subset of IDs within 60-second windows are presented in Table 6.3a, while Table 6.3b shows the corresponding ID ratios for the same time windows. These ratios were obtained by dividing all values by their respective row minimums. For instance, in the time window (0,60), the row minimum is 40, which corresponds to the number of IDs for ID 277. To compute the ratios in Table 6.3b, all values in the time window (0,60) are divided by 40. This ratio relationship holds true for all cluster IDs across all windows. In the case of a poisoning attack, maintaining this consistent relationship would require the adversary to manipulate all IDs simultaneously with the same ratio. However, this is challenging due to the CAN ID-based priority mechanism. Exploiting this property, we employ it to detect data poisoning attacks within CAN data sequences during streaming learning. To achieve this, we conducted experiments using three outlier detection techniques: sum of ratio-based, Mahalanobis distance-based, and autoencoder-based outlier detection. These methods aim to identify deviations from the expected ID ratios and detect anomalies in the data. Prior to applying these techniques, max-min normalization is applied on each ID to mitigate biases towards high ratio values.

Table 6.3: ID counts and ratios for 60 seconds observation windows for a subset of IDs

(a) ID counts

Time window	ID 006	ID 00E	ID 03A	ID 277	ID 075	ID 130
(0, 60)	60	5993	120	40	120	598
(60, 120)	60	5999	119	40	120	600
(120, 180)	36	3600	73	24	72	370
(180, 240)	38	3600	74	24	71	375
(240, 300)	58	6000	120	40	120	600

(b) ID ratios

Time window	ID 006	ID 00E	ID 03A	ID 277	ID 075	ID 130
(0, 60)	1.50	149.83	3.00	1.00	3.00	14.95
(60, 120)	1.50	149.98	2.98	1.00	3.00	14.95
(120, 180)	1.50	150.00	3.04	1.00	2.96	15.42
(180, 240)	1.58	150.00	3.08	1.00	2.96	15.63
(240, 300)	1.45	150.00	3.00	1.00	3.00	15.00

Sum of Ratio-based Outlier Detection

Given the consistent ratio values for IDs across all windows, deviations from the expected sum of ratios within a window can indicate anomalies. Anomalies are detected using the interquartile range (IQR), a dispersion measure capturing the range between the first quartile ($Q1$) and third quartile ($Q3$) of the data [203]. Anomalies are identified as observations falling below $Q1 - 1.5 \times IQR$ or above $Q3 + 1.5 \times IQR$. This measurement is a common approach in statistics for detecting outliers and anomalies [203, 204]. These threshold values are identified using benign datasets.

Mahalanobis Distance-based Outlier Detection

Clustered IDs with normalized ratio values create a multivariate distribution characterized by correlated values. Therefore, Mahalanobis distance is an effective technique to identify anomalous ratio values. For each cluster, mean vectors and inverse covariance matrices are calculated using 80% of benign data. The remaining 20% of data is utilised to determine the thresholds using the IQR method. If any clusters identify a given sample as anomalous, it is classified as a poisoned data sample. Conversely, if none of the clusters exceeds the defined thresholds, the sample is declared benign.

Autoencoder-based Outlier Detection

Autoencoders are a reliable approach for detecting anomalies within multivariate distributions. In this case, individual autoencoders are trained for each cluster using the normalized ratio values as input. Similar to the Mahalanobis distance-based method, we use 80% train and 20% test split for training the autoencoders and estimating the thresholds. The reconstruction error is used to compute the anomalous thresholds based on the IQR. Subsequently, anomalous samples are identified using the same threshold-based approach as in the Mahalanobis distance-based method.

6.5.3 Data Poisoning Defending Procedure

Based on the successful data poisoning attack detection techniques from our experiments, we incorporate a filtering step to CAN-ODTL model to remove poisonous samples prior to retraining. This defence mechanism, referred to as defending before training, aims to sanitize the data before the training process [205]. Among the tested outlier detection methods, this utilises the Mahalanobis distance-based outlier detection due to its capability to identify even minor instances of data poisoning attacks within a short period of time. This procedure is shown in Algorithm 4.

This algorithm processes the streaming CAN data by extracting CAN IDs and timestamps from the frames. It keeps track of the minimum time (t_min) in each window to monitor the progression of time. It analyzes the data within a time window T to determine if the samples within that window are potentially poisonous. Defending against data poisoning attacks similar to the fuzzing attacks can be done by removing the new CAN IDs. ID dictionary is maintained to keep track of the counts for each valid ID for time T. After analyzing the messages within the time window, the ID dictionary is divided into pre-identified clusters, and the Mahalanobis distance M_d is calculated for each cluster. These distances are then compared to predefined thresholds, and if any of the distances exceed the thresholds, the observed ID sequence is considered as poisoned and is removed. If the sequence is determined to be benign, it is used to retrain the classification layer of the pre-trained CAN-ODTL model. By identifying and removing poisonous data, the algorithm helps ensure the integrity of the retraining process and improves the model's ability to accurately detect anomalies in the CAN data. Moreover, if the observed poisonous behaviour is a result of technical faults rather than adversarial actions, one of the advantages of this proposed method is to enhance the system resilience.

Algorithm 4 Data Poisoning Defending

Input: Pre trained model M, Streaming CAN data F, Valid ID list L, ID clusters C, Anomaly thresholds ω_1, ω_2, Mean vectors $\hat{\mu}$, Inverse of covariance matrices S^{-1}, Time window T
Output: Benign ID sequence D
Init: $ID\_dictionary$ = {id:count}, D = [ ] ▷ $id \in L$
while F is not empty **do**
  Read l in F ▷ Read line by line in F
  Pre-process l,
  Extract id, time_stamp t, t_min
  while $t - t\_min \leq T$ **do**
    if $id \notin L$ **then**
      delete id ▷ Defence for fuzzing attacks
    else
      Update $ID\_dictionary$
      Append id to D
    end if
  end while
  for $c \in C$ **do**
    Pre-process $ID\_dictionary$
    Calculate M_d
  end for
  if any $\omega_1 \geq M_d \geq \omega_2$ **then** ▷ Minimum and maximum thresholds
    delete D
  else
    return D
  end if
  Retrain $M(D)$ ▷ CAN-ODTL model retraining
  $t\_min \leftarrow t$
end while

6.6 Evaluation and Performance Results - Preventing Data Poisoning Attacks

This section discusses the dataset used for the experiments, the creation of poisoned data, ID count behaviour during the data poisoning attack, model performance after training with poisoned data, data poisoning attack detection, limitations of this work, and the analysis of memory usage and training time.

6.6.1 Dataset

This work utilises the ROAD dataset [30] and we select the attacks that target the CAN ID 0D0 and 6E0 as these include eight different attacks with sufficiently large datasets. We assume that the pre-trained model is trained on a benign dataset that does not contain any poisoned samples. Furthermore, we assume that the number of ECUs in a vehicle, represented by CAN IDs in our model, remains fixed, and we have observed all the CAN IDs within the dataset used to train the initial model.

6.6.2 Experimental Setup

We use the CAN-ODTL model architecture discussed in the Section 6.3. To identify clusters, hierarchical clustering is performed on the pairwise correlation matrix using Ward's method. Accordingly, IDs are grouped into clusters based on their highest correlation coefficients. This resulted in three clusters, where two clusters exhibited over 99% correlated IDs, and one cluster included IDs with correlations ranging between 94-98%. The cluster sizes are 94, 7, and 5 IDs, respectively. The autoencoder model includes 32 nodes and a latent layer with 10 nodes, and grid search is used to determine the optimal hyperparameters. Experiments were run on Raspberry Pi 4 Model B 8GB version with a 16GB micro SD card for the overhead analysis. Tensorflow 2.10 and Python 2.8 is installed in Raspberry Pi for the model retraining.

6.6.3 CAN ID Count Change During Benign Driving

Generally, non-diagnostic messages in ECUs are commonly broadcasted at regular intervals [9, 206]. However, our analysis reveals minor fluctuations in the number of IDs within specific time windows. This variability is clearly observed in Figure 6.5, illustrating the average number of frames per second for different benign datasets from the ROAD dataset. Notably, the extended, all bits, and benign anomaly datasets show a lower average number of frames. Further analysis of the benign anomaly dataset's ID transitions helps identify the reasons behind this observation, as depicted in Figure 6.6. The analysis indicates that for some IDs, such as '419' and '533', the inter-arrival time changes during specific intervals, while for others, like '03C' and '1C4', the inter-arrival times remain consistent throughout the duration. This variability might be attributed to contextual dependencies or anomalous benign activities for certain IDs. Table 6.3a provides additional insights into observed ID count changes within one-minute time windows. To explore this observation across different vehicles, we collected several hours of benign datasets from two additional car models. Figure 6.7 illustrates variations in ID

counts for these vehicles within one-minute intervals. The observed drops in ID counts are not due to interruptions in the CAN ID streams, as the data was consistently recorded throughout the collection period. As shown in Figure 6.7b, the IDs associated with another cluster of the car 2 Nero remain steady throughout the collection period. These changes in some CAN IDs result in the generation of a large number of ID sequences under diverse driving scenarios. Since these fluctuations are not a result of attacks, it is crucial to incorporate these behaviours into IDS training to minimize false positives. Despite the fluctuations in ID counts, the ratios of the ID counts remain relatively constant within the same ID cluster.

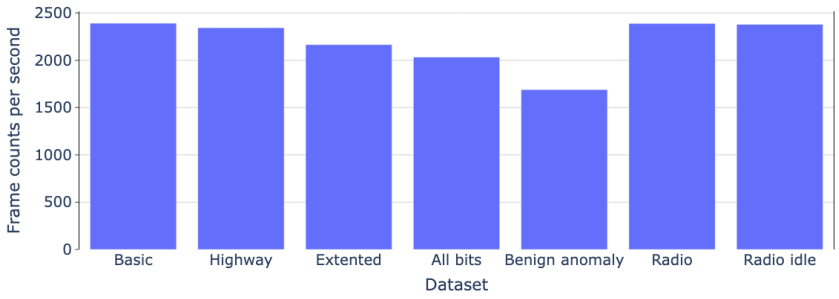


Figure 6.5: Average frame counts per second for different benign datasets

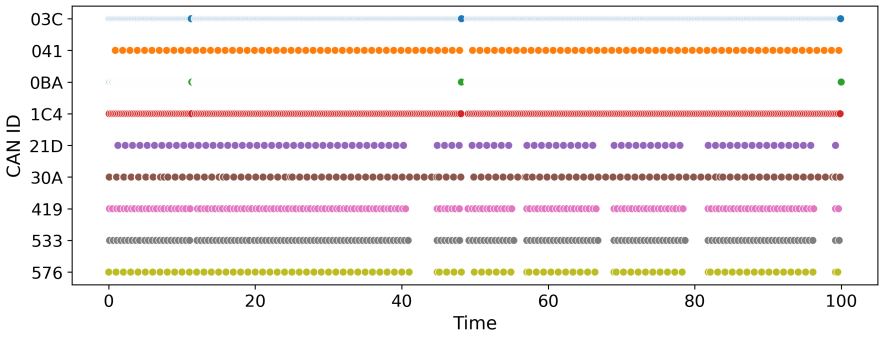
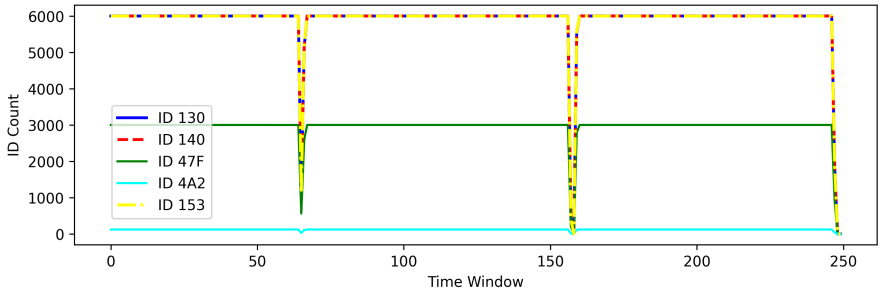


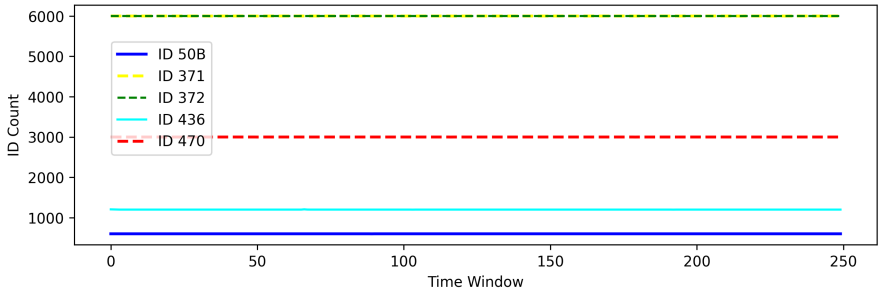
Figure 6.6: ID frames transmission during a 100s period in the benign anomaly dataset

6.6.4 Poisoned Data Creation

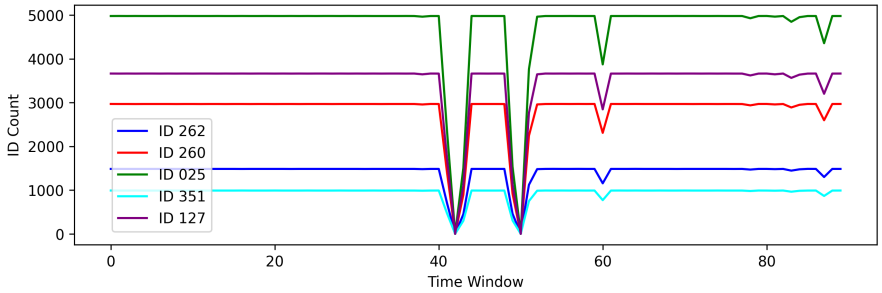
To create poisoned datasets, we analysed the behaviour of ID counts during seven realistic injection attacks of the ROAD dataset. In order to generalize our findings, HCRL CH [85] and HCRL SA [175] are also used for the analysis.



(a) Car 1 ID counts change - Cluster 1



(b) Car 1 ID counts change - Cluster 2

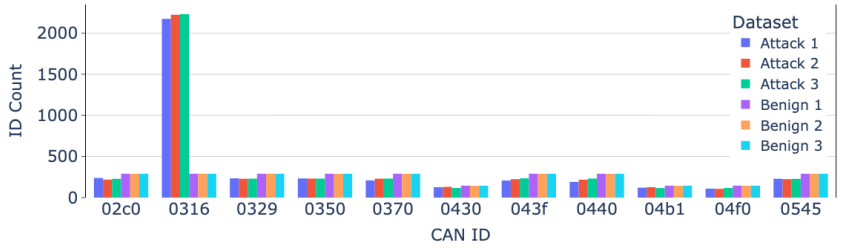


(c) Car 2 ID counts change

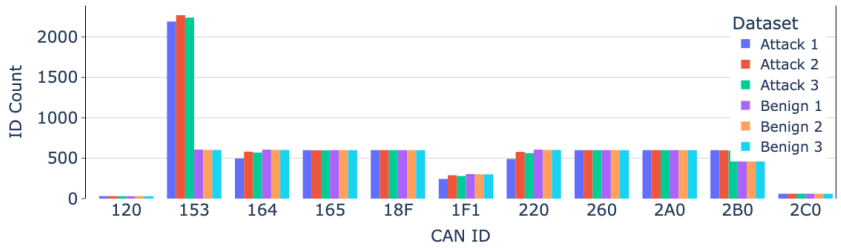
Figure 6.7: ID count change for different one minute time windows. Each subfigure represents only a subset of the CAN IDs which are in same clusters

Figure 6.8c shows the change of ID counts during the benign driving and reverse light on attack of the ROAD dataset. It can be observed that while there are highly correlated IDs, the targeted ID attacks do not impact the ID counts of other correlated or uncorrelated IDs during the attack period. This behaviour is consistent across all injection attacks in the three datasets including ROAD, HCRL CH and HCRL SA as shown in Figure 6.8a and Figure 6.8b. Given our threat model, which assumes that data poisoning is similar to data injection in this context, we randomly injected targeted IDs with a percentage $P\%$ to create poisoned data that maintains the same characteristics as the realistic attack

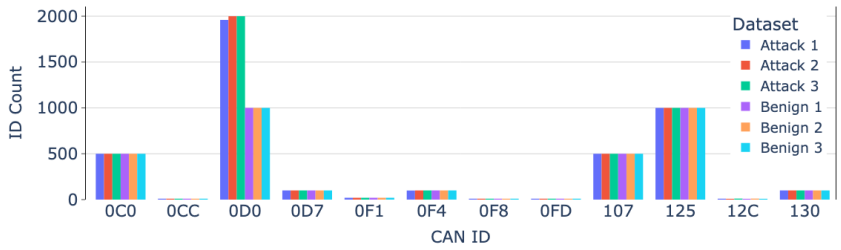
data. Adversaries may attempt to inject IDs immediately after the appearance of the targeted legitimate ID or based on a more elaborate strategy. However, our analysis of the ROAD dataset reveals that these injections occur in a random order despite the attack injection strategy, which may be attributed to the CAN ID-based priority mechanism. Additionally, since we consider the ID count within a fixed time window as an indicator of detecting poisoned data, regardless of the method used for ID injection during poisoning, the algorithm remains effective in detecting data poisoning due to its ability to identify changes in ID counts.



(a) HCRL CH RPM attack



(b) HCRL SA malfunction attack



(c) ROAD reverse light on attack

Figure 6.8: ID counts change during the benign driving and driving under injection attacks. Each includes three benign and attack data samples obtained within a fixed time window. This represents only a subset of the CAN IDs for each dataset

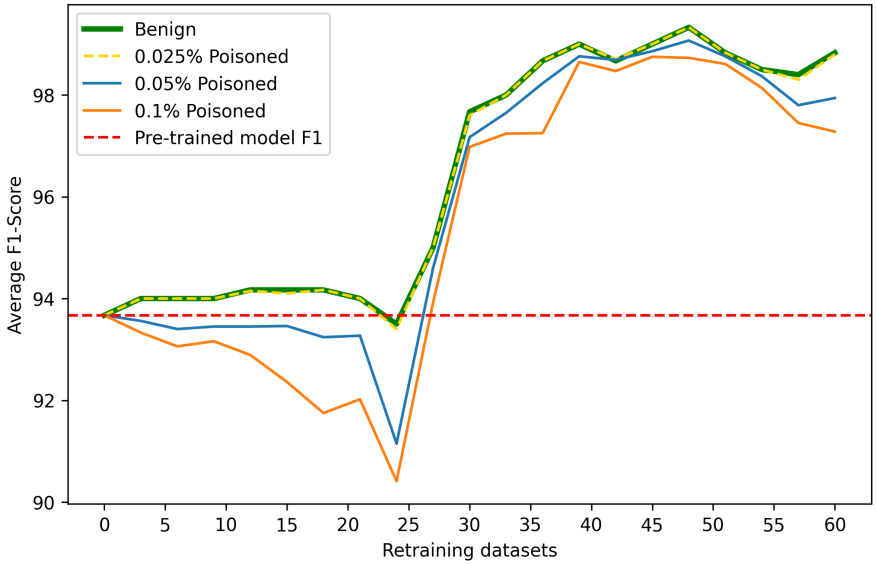
6.6.5 Model Retraining with Poisoned Data

We evaluated the attack detection capability of the CAN-ODTL based IDS by retraining it with benign (non-poisoned) and varying percentages of poisoned datasets. The benign dataset, consisting of one hour of driving data, was divided into 60 subsets, each representing 60 seconds of data. During retraining with benign data, the subsets were used without any modifications, while for poisoned retraining, the targeted IDs 0D0 and 6E0 were injected with percentages ranging from 0.01% to 0.5%. A one-minute data subset typically contained around 143,460 frames, with 0.01% poisoning resulting in approximately 14 injected frames, and 0.5% poisoning resulting in around 717 injected frames.

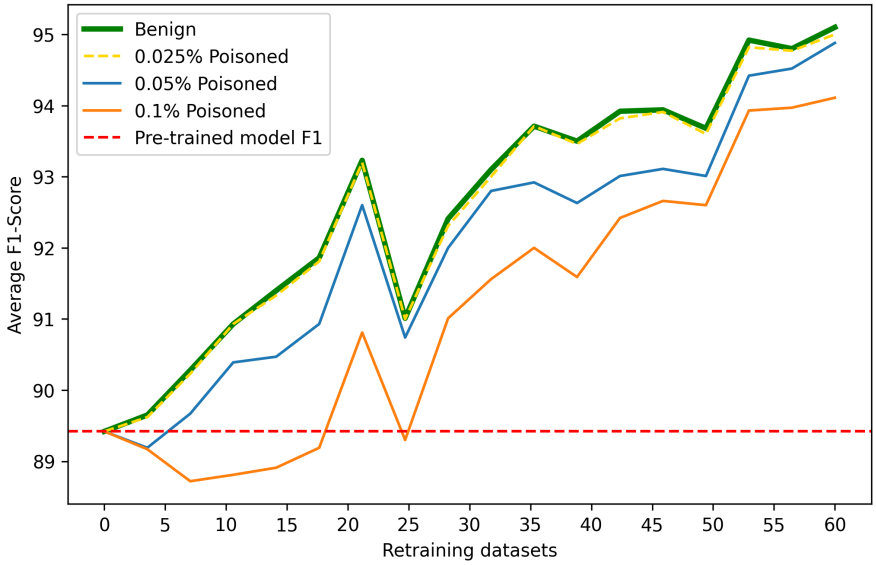
To evaluate attack detection performance, we consider injection and masquerade attacks on ID 0D0 and ID 6E0. ID 0D0 attacks include max speedometer, reverse light on, reverse light off, and their masquerade counterparts. ID 6E0 attacks include correlated signal and its masquerade version. In Figure 6.9a, the attack detection capability is shown for ID 0D0 attacks with various ID 0D0 data poisoning percentages. The red dashed line represents the pre-trained model's F1-score, which is 93.67%. Retraining with benign datasets slightly improves, with a drop in F1-score at the 25th dataset. However, significant progress occurs thereafter, reaching a final F1-score of 98.83% after the 60th dataset. At a 0.1% poisoning percentage, the model's performance initially drops significantly until the 25th dataset but then improves to a 97.28% F1-score, resulting a 1.55% reduction compared to benign retraining. The same results for ID 6E0 attacks are shown in Figure 6.9b. Benign retraining exhibits a 3.81% F1-score improvement until the 21st dataset, with a slight drop until the 25th dataset. Subsequently, there is progress, resulting in a 5.67% improvement over 60 datasets. Similar to ID 0D0 data poisoning, 0.05% and 0.1% ID 6E0 data poisoning reduces the model's attack detection capability. Notably, for both IDs, at a poisoning percentage of 0.025%, the detection capability remains at the same level as the benign retraining across all datasets, indicating that 0.025% poisoning is insufficient to degrade the model's retraining effectiveness. Benign retraining highlights that certain data samples significantly contribute to the model's progress while others slightly downgrade its performance. This behaviour may be attributed to the emergence of new CAN ID sequences in these datasets. Therefore, it is important to pre-train the model with a sufficiently large dataset and retrain for a longer period until it trains with a larger number of sequences. Additional experiments conducted with higher poisoning percentages demonstrate a decrease in performance as the poisoning percentage increases. The retraining process with more than 0.5% poisoning of ID 0D0 data is depicted in Figure 6.10. Similar outcomes are observed for ID 6E0

data poisoning.

In addition to the high frequent ID 0D0 and 6E0, we further explore data poisoning with other IDs, 0C0 (medium frequent) and 4E7 (low frequent) which were selected randomly representing each cluster. The results show that injecting at least 0.1% of these IDs is necessary to downgrade the model's performance against attacks targeting the ID 0D0 and 6E0. For poisoning attacks below 0.1% poisoning, the model's performance remains comparable to that of benign dataset retraining. This highlights the effectiveness of targeted poisoning when the same ID is employed during the CAN bus attack. Similar observations in federated learning research [202] confirm that targeted attacks significantly impact the attacked classes while minimally affecting the remaining classes.



(a) Data poisoning with ID 0D0 for attacks that target ID 0D0



(b) Data poisoning with ID 6E0 for attacks that target ID 6E0

Figure 6.9: Retraining progress with different targeted data poisoning percentages. X-axis represents 60 retraining samples which each includes one minute CAN frames

To assess the impact of poisoned data on other CAN ID-based IDSs, we select the IDS proposed in [200], which utilised an LSTM model to capture the patterns within CAN ID sequences. This model predicted the next CAN ID based on a sequence of the preceding

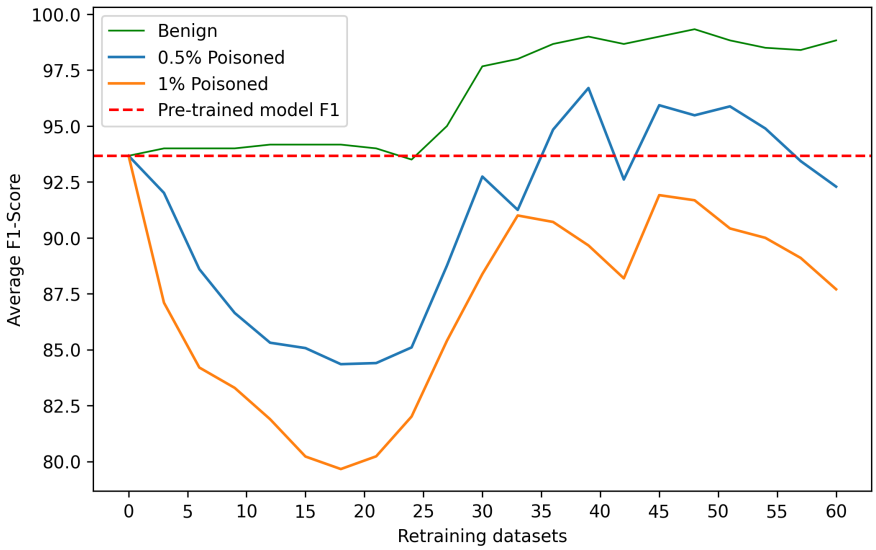
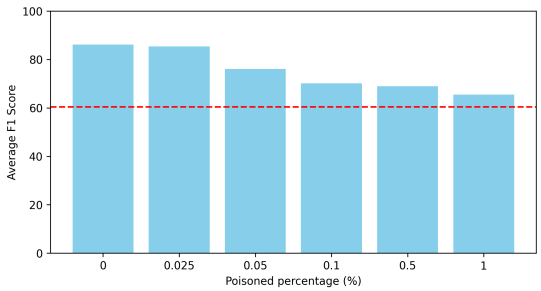


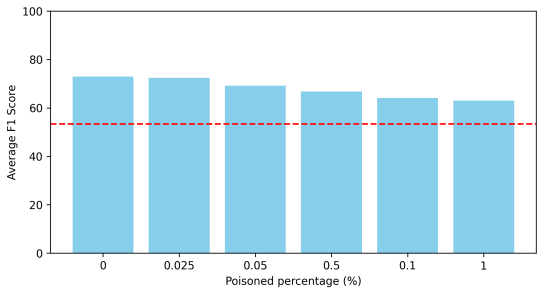
Figure 6.10: Retraining progress with over 0.5% ID 0D0 data poisoning percentages. X-axis represents 60 retraining samples which each includes one minute CAN frames

20 CAN IDs. The predicted ID is then compared with the actual ID to detect anomalies. The architecture of their model includes two dense layers with 128 nodes, two LSTM layers with 512 nodes, dropout layers, and a final dense layer with a softmax activation function. Instead of ID-based log losses, a common log loss was employed as the anomaly signal in their setup. Notably, this model exhibits greater complexity than the lightweight CAN-ODTL model. As this model is not designed for streaming learning, our approach initially involved training the model with the initial benign dataset, which was used to pre-train the CAN-ODTL model. Subsequently, 60 samples from the poisoned dataset were combined to create a single poisoned dataset, maintaining their sequential order. This poisoned dataset was then merged with the benign pre-training dataset to assess the model's performance in the presence of poisoned data. This process was performed individually for each poisoned percentage. To evaluate the model's performance with a large benign dataset, 60 benign samples were integrated with the initial benign dataset. The performance in detecting attacks under varying poisoning percentages for ID 0D0 and 6E0 is depicted in Figure 6.11. The red dashed lines represent the average F1-score obtained after training with the initial dataset. Notably, employing a larger benign dataset (0% poisoned) during training contributes to achieving higher F1-scores for both types of ID attacks. Similar to the CAN-ODTL model's behaviour, the effectiveness of attack detection diminishes with higher percentages of data poisoning during training.

Comparatively, this model's ability to detect attacks is relatively lower than that of the CAN-ODTL model for both attack scenarios. These experiments emphasize that training CAN ID-based IDSs with poisoned data results in a degradation of model performance.



(a) Data poisoning with ID 0D0 for attacks that target ID 0D0



(b) Data poisoning with ID 6E0 for attacks that target ID 6E0

Figure 6.11: Baseline model attack detection capability with different percentages of data poisoning. The red dashed line represents the F1-score for the initial training dataset

6.6.6 Data Poisoning Attack Detection

Given the impact of even a small percentage of data poisoning attacks on the model retraining process, we compare the performance of three different anomaly detection methods: the sum of ratio, Mahalanobis distance, and autoencoder-based detection to detect data poisoning attacks. For evaluation, we utilise 60 benign datasets and 60 poisoned datasets with varying percentages of poisoning. In addition to targeting the ID 0D0 and 6E0, we randomly select two additional CAN IDs, 0C0 and 4E7 to poison the benign data, representing two different clusters. The evaluation of these poisoned attacks is presented in Table 6.4, which shows the accuracy (Acc), false positive rate (FPR), and false negative rate (FNR) of the detection methods for the poisoned IDs at different percentages of poisoning.

The Mahalanobis distance-based method consistently outperforms the sum of ratio and autoencoder-based methods across all IDs and poisoned percentages. It demonstrates excellent detection capabilities with 0 FPR, even at low poisoning percentages such as 0.05%. Although it failed to detect 0.025% poisoned samples that do not degrade the IDS's performance, it maintained a 0 FPR to facilitate retraining with benign datasets. Furthermore, the Mahalanobis distance-based method successfully detected 0.025% poisoned attacks targeting the medium frequent ID 0C0 and the low frequent ID 4E7. This can be attributed to the lower variability exhibited by medium and low frequent IDs

compared to high frequent IDs like 0D0 and 6E0. When targeting multiple IDs simultaneously, the detection performance is further improved. This is due to the increased deviation from the expected ratios of multiple IDs, making the data poisoning attacks more easily detectable.

The sum of ratio and autoencoder-based methods perform similarly, with the autoencoder-based method slightly better for low percentage attacks. However, both methods have higher FPR. The FPR remains consistent across all IDs for both the sum of ratio and autoencoder-based models. This is attributed to the misclassification of the same benign samples according to their attack detection thresholds. Increasing the training dataset size may improve the performance of the autoencoder-based method.

In addition to ID injection, the adversary could employ the ID suspension attack to poison the training data, effectively reducing the number of CAN IDs within a specific time period. Despite the absence of suspension attacks in the ROAD dataset, we evaluated the effectiveness of the proposed data poisoning attack detection against such manipulations. This was achieved by removing targeted IDs for varying time intervals. Since our data samples were one-minute streaming sequences, we conducted experiments with suspension attacks lasting 0.01, 0.1, 1, 15, 30, and 60 seconds. The results are presented in Table 6.5, showcasing the data suspension attack detection for suspension times of 0.01, 0.1, 1, and 15 seconds across different suspension IDs. The results show the effectiveness of the Mahalanobis distance-based method in detecting even short suspension attacks of 0.1 seconds duration. The results for 30 and 60 seconds are similar to those obtained for 15 seconds, across all IDs and attack detection methods. This only fails to detect 0.01 seconds suspension attack, which might not significantly impact the model's performance. The autoencoder-based method exhibits slightly better performance compared to the sum of ratio method. ID 4E7 is a low-frequency ID with intervals of over 0.1 seconds between two consecutive frames. Therefore, a suspension attack of over 1 second is considered for ID 4E7. In contrast, the Mahalanobis distance-based method proves effective in detecting even small percentages of data poisoning attacks. While the adversary is not constrained to a specific data poisoning percentage, our findings indicate that higher poisoning percentages are more easily detectable. As a result, we focused our experiments on data poisoning up to 1% for the analysis.

6.6.7 Limitations

While the proposed approach demonstrates superior capabilities in detecting data poisoning, it is important to acknowledge its limitations. One potential limitation is the possibility of inadvertently excluding some benign data samples that exhibit significant differences from other benign sequences. The process of normalizing ID ratios for each CAN ID demands accurate identification of the true minimum and maximum ID counts for each ID. Failure to achieve this could lead to increased false positives or negatives. However, employing a large dataset containing diverse benign driving data helps minimize this limitation.

The proposed approach relies on the ratios of ID counts within the same clusters. Consequently, adaptive attackers who are aware of this defense strategy may attempt to bypass it by injecting or suspending IDs while maintaining the same ratios. To achieve this, the adversary would need to inject or suspend all IDs within a specific cluster using a consistent multiplication factor. Given the ID-based priority mechanism in the CAN bus, it becomes difficult for the adversary to maintain uniform ratios for all IDs. This challenge arises from the fact that lower-priority IDs must wait for higher-priority IDs to access the bus. As a result, achieving uniform ratios becomes highly challenging for the adversary. Nonetheless, detecting such attempts is possible by identifying the maximum number of ID counts for a cluster within a one-minute timeframe. Counting this during training can help detect such efforts. However, accurately identifying these maximum counts necessitates a well-represented sample from the benign driving dataset, as low maximum counts could inadvertently exclude certain benign data samples.

6.6.8 Memory Usage and Training Time Analysis

To optimize memory usage and retraining speed on the resource-constrained Raspberry Pi, minimizing computational overhead is crucial. Data poisoning detection prior to retraining requires additional computations, such as counting CAN IDs and calculating the Mahalanobis distance. It also involves storing an ID counting dictionary, mean vectors, inverse covariance metrics for each cluster, and one minute of streaming CAN IDs in memory. With the CAN-ODTL based IDS using 157MB of memory during retraining, an additional 23MB is needed for these data, well within the 8GB available memory of the Raspberry Pi. While the CAN-ODTL processes 60 seconds of data in 90 seconds, the Mahalanobis distance-based data poisoning detection adds approximately 8 seconds, totalling 98 seconds. Considering that the CAN-ODTL based IDS does not perform

Table 6.4: Data poisoning attack detection performance - ID injection

ID	Poisoned %	Sum of ratio			Mahalanobis			Autoencoder		
		Acc	FPR	FNR	Acc	FPR	FNR	Acc	FPR	FNR
0D0	0.01	50.0	21.6	78.3	50.0	0.0	98.3	50.0	18.3	78.3
	0.025	50.0	21.6	78.3	50.0	0.0	98.3	50.0	18.3	78.3
	0.05	83.3	21.6	11.6	**100.0**	0.0	0.0	88.3	18.3	1.6
	0.1	88.3	21.6	1.6	100.0	0.0	0.0	89.1	18.3	0.0
	0.2	88.3	21.6	1.6	100.0	0.0	0.0	89.1	18.3	0.0
	0.3	89.1	21.6	0.0	100.0	0.0	0.0	89.1	18.3	0.0
0C0	0.01	48.3	21.6	81.6	72.5	0.0	51.6	50.0	18.3	78.3
	0.025	83.3	21.6	11.6	**100.0**	0.0	0.0	84.1	18.3	10.0
	0.05	87.5	21.6	3.3	100.0	0.0	0.0	88.3	18.3	1.6
	0.1	88.3	21.6	1.6	100.0	0.0	0.0	89.1	18.3	0.0
	0.2	89.1	21.6	0.0	100.0	0.0	0.0	89.1	18.3	0.0
	0.3	89.1	21.6	0.0	100.0	0.0	0.0	89.1	18.3	0.0
6E0	0.01	50.0	21.6	78.3	50.0	0.0	98.3	50.0	18.3	78.3
	0.025	50.0	21.6	78.3	50.0	0.0	98.3	50.0	18.3	78.3
	0.05	50.0	21.6	78.3	**100.0**	0.0	0.0	50.0	18.3	78.3
	0.1	88.3	21.6	1.6	100.0	0.0	0.0	89.1	18.3	0.0
	0.2	89.1	21.6	0.0	100.0	0.0	0.0	89.1	18.3	0.0
	0.3	89.1	21.6	0.0	100.0	0.0	0.0	89.1	18.3	0.0
4E7	0.01	48.3	21.6	81.6	50.8	0.0	98.3	50.0	18.3	78.3
	0.025	88.3	21.6	1.6	**100.0**	0.0	0.0	89.1	18.3	0.0
	0.05	89.1	21.6	1.6	100.0	0.0	0.0	89.1	18.3	0.0
	0.1	89.1	21.6	0.0	100.0	0.0	0.0	89.1	18.3	0.0
	0.2	89.1	21.6	0.0	100.0	0.0	0.0	89.1	18.3	0.0
	0.3	89.1	21.6	0.0	100.0	0.0	0.0	89.1	18.3	0.0
0D0,0C0	0.01	48.3	21.6	81.6	72.5	0.0	51.6	50	18.3	78.3
	0.025	83.3	21.6	11.6	**100.0**	0.0	0.0	84.1	18.3	10.0
	0.05	87.5	21.6	3.3	100.0	0.0	0.0	88.3	18.3	1.6
	0.1	89.1	21.6	1.6	100.0	0.0	0.0	89.1	18.3	0.0
	0.2	89.1	21.6	0.0	100.0	0.0	0.0	89.1	18.3	0.0
	0.3	89.1	21.6	0.0	100.0	0.0	0.0	89.1	18.3	0.0

Table 6.5: Data suspension attack detection performance - ID suspension

ID	Suspension time (s)	Sum of ratio			Mahalanobis			Autoencoder		
		Acc	FPR	FNR	Acc	FPR	FNR	Acc	FPR	FNR
0D0	0.01	50.0	21.6	78.3	56.6	0.0	86.6	50.0	18.3	81.6
	0.1	88.3	21.6	1.6	**100.0**	0.0	0.0	90.8	18.3	0.0
	1	89.1	21.6	0.0	100.0	0.0	0.0	90.8	18.3	0.0
	15	89.1	21.6	0.0	100.0	0.0	0.0	90.8	18.3	0.0
0C0	0.01	50.0	21.6	78.3	66.6	0.0	66.6	50.0	18.3	81.6
	0.1	50	21.6	78.3	**100.0**	0.0	0.0	50.0	18.3	81.6
	1	89.1	21.6	0.0	100.0	0.0	0.0	50.0	18.3	81.6
	15	90.2	21.6	0.0	100.0	0.0	0.0	91.6	18.3	0.0
6E0	0.01	50.0	21.6	78.3	50.0	0.0	100.0	50.0	18.3	81.6
	0.1	50.0	21.6	78.3	**100.0**	0.0	0.0	50.0	18.3	81.6
	1	90.2	21.6	0.0	100.0	0.0	0.0	91.6	18.3	0.0
	15	90.2	21.6	0.0	100.0	0.0	0.0	91.6	18.3	0.0
4E7	1	89.1	21.6	0.0	**100.0**	0.0	0.0	90.8	18.3	0.0
	15	89.1	21.6	0.0	100.0	0.0	0.0	90.8	18.3	0.0
0D0,0C0	0.01	50.0	21.6	78.3	60.8	0.0	78.3	50.0	18.3	81.6
	0.1	89.1	21.6	0.0	**100.0**	0.0	0.0	90.8	18.3	0.0
	1	89.1	21.6	0.0	100.0	0.0	0.0	90.8	18.3	0.0
	15	89.1	21.6	0.0	100.0	0.0	0.0	90.8	18.3	0.0

inference during retraining, using Mahalanobis distance-based anomaly detection is well-suited for monitoring CAN data and identifying potential data poisoning attacks before retraining. The proposed solution can be deployed on Raspberry Pi 4, serving as a separate ECU mounted to the CAN bus, ensuring that retraining is performed using non-poisoned data and enhancing the integrity of the retraining process.

6.7 Conclusion

One-class classification-based CAN IDSs, although effective, necessitate a large dataset of benign frames to mitigate the impact of unseen benign frames. However, collecting a comprehensive and diverse dataset that adequately reflects various benign driving behaviours is challenging, and training a deep learning model with such a dataset can be computationally expensive. In response to these challenges, this chapter introduced CAN-ODTL, an on-device transfer learning technique designed to retrain the classification layer of a GRU-based IDS using CAN streaming data on a resource-constrained Raspberry Pi device. Experimental results based on the ROAD dataset demonstrate that the detection capability of attacks can be enhanced by selecting a larger and more diverse benign dataset. Furthermore, the study underscores the significance of a pre-trained model trained on a substantial dataset for effective model retraining. A comparative analysis between CAN-ODTL retrained with a larger dataset, and a pre-trained model using only a fraction of this dataset indicates that retraining can achieve improved detection with a lower false positive rate. Overhead analysis emphasizes the efficiency of CAN-ODTL, making it a viable option for deployment in real vehicles. This approach not only enhances attack detection but also optimizes computing resources.

However, the CAN-ODTL procedure is vulnerable to data poisoning attacks, which can significantly degrade IDS performance. To address this vulnerability, this chapter introduces a Mahalanobis distance-based anomaly detection method for data sanitization before retraining. Given the absence of datasets for evaluating the impact of data poisoning attacks on CAN IDS, datasets with varying percentages of poisoned data were generated to simulate realistic data poisoning attack behaviours. Experimental results illustrate the degradation of IDS performance corresponding to the percentage of poisoned data and highlight the effectiveness of the proposed method in detecting even small percentages of data poisoning attacks. Furthermore, the feasibility of deploying this method on Raspberry Pi for monitoring CAN data before retraining is validated, ensuring data integrity during the retraining process.

Chapter 7

Improved Autoencoder-based IDS for CAN Payload data

The CAN ID-based model, introduced in Chapter 5, and an enhanced version incorporating streaming learning capabilities, as presented in Chapter 6, solely leverage the CAN ID field to generate features. This includes CAN-ID sequences utilised by the GRU-based model and inter-arrival times employed by the time-based model in the CAN-CID model. While the time-based model exhibits promising results for injection attacks, the GRU model demonstrates the ability to detect both injection and masquerade attacks. Masquerade attacks present in the ROAD dataset introduce slight time shifts compared to legitimate ECU frame transmissions. Such subtle time-synchronization mismatches in masquerade attacks, including scenarios where no frames from the legitimate ECU are transmitted for a fraction of the time while initiating masqueraded attacks, can be detected by the GRU-based model. However, sophisticated attackers with expert hacking knowledge might employ advanced masquerade attacks that leave the sequences unaltered. This makes ID-based models ineffective in detecting such attacks. Instead, such attacks necessitate an IDS that utilises the payload field. Therefore, to detect these sophisticated masquerade attacks, this chapter presents an improved AE-based IDS utilising the time-series CAN payload data. This chapter addresses the RQ3.

The main findings of this chapter were published in the Journal of Information Security and Applications, 2023 [32].

7.1 Introduction

The CAN payload field supports data transmission of up to 64 bits, facilitating the exchange of diverse information among different ECUs [207]. The prevalence of a large number of ECUs in modern vehicles results in multivariate time-series data for each CAN ID. However, the lack of knowledge regarding the CAN data specifications, stored in the CAN DBC file, poses challenges in developing payload-based IDSs for widespread adoption. Achieving complete accuracy for certain attacks is considered unrealistic [97]. Payload-based IDSs typically employ two approaches [30]. The first approach, known as the black-box approach, treats the data frame as a string of bits without decoding the signals they represent. The second approach involves decoding the raw data field into constituent signals and using the identified signal values as inputs. While a few IDSs utilise the encoded signals as inputs [100, 66, 96], the majority of payload-based IDSs [99, 101, 103, 102, 104, 106, 97, 98] adopt the black-box approach. A meta-analysis of several papers [34] also indicates that most payload-based IDSs use the payload field without decoding it into signals. IDSs leveraging decoded signal values typically rely on CAN DBC files or reverse engineering approaches. Although these models achieve higher attack detection rates by selecting only relevant signals, they are not vehicle-agnostic [208]. Conversely, IDSs using the black-box approach face challenges of low attack detection rates and high computational resource requirements, as they consider all features of the payload field. Consequently, there is a clear need for a vehicle-agnostic and lightweight payload-based model to effectively detect sophisticated attacks, such as masquerade attacks.

AE-based IDSs are employed for anomaly detection across various application domains, including IVNs [101, 102, 130, 209, 100]. The conventional assumption is that reconstruction errors are significantly higher for anomalous samples compared to benign ones. However, vanilla AEs are susceptible to overgeneralization [45], leading to the reconstruction of anomalous inputs without elevated reconstruction errors and resulting in numerous false negatives. This poses a critical issue for certain cybersecurity problems, such as IVN security. To mitigate this drawback, several enhanced AEs have been proposed in the literature. In [45], the authors introduced a memory-augmented AE by incorporating a memory module into the vanilla AE. This memory module stored the prototypical elements of normal data during model training. Instead of the latent vector generated by the encoder, the most relevant memory items of the latent vector were utilised as input to the decoder. However, this approach requires separate memory modules when

normal datasets exhibit different groups, such as CAN IDs in CAN data, leading to varying data patterns. Moreover, some datasets may require substantial memory to store prototypical elements [210]. Another method, presented in [211], utilised latent space distribution to detect anomalies. However, employing k nearest neighbor calculations demands higher computational resources for successful anomaly detection. In [212], a series of multi-layer perceptrons (MLPs) were employed in the latent space to predict each latent space element, and the predicted array was then used as input to the decoder. Nonetheless, none of these approaches are suitable for deployment in IVNs for anomaly detection due to limited computational resources. To alleviate this problem and address the aforementioned challenges, this chapter proposes a novel AE-based IDS utilising the CAN payload data.

7.2 Chapter Contribution

The proposed AE model modifies the latent space of the vanilla AE with a novel feature selection technique to identify cyberattacks on the CAN payload data in near real-time. The main contribution of this chapter can be summarized as follows:

1. Proposes an improved AE model to detect both point and contextual anomalies in the CAN bus. The proposed AE model, named as Latent AE, exploits the association of CAN payload variables to identify anomalies. Latent AE addresses the issue of overgeneralization of vanilla AEs for anomaly detection in the CAN bus.

2. Latent AE is a lightweight model; Cramér's statistic-based [213] feature selection technique reduces the complexity of the data by removing weakly associated features. This helps improve the model's computation efficiency and accuracy due to the removal of noise data. Feature selection avoids the need for a large dataset compared to the dataset requirement for a model with all variables.

3. Latent AE is efficient and can be deployed in resource-efficient edge devices. The proposed data structure incorporates CAN IDs into the payload variable, allowing the use of only one model for all CAN IDs without building separate models for each CAN ID.

Table 7.1: Description of SynCAN attack datasets

Attack	Description
Plateau	Change signal value into a constant value
Continuous	Change signal value so that it slowly drifts away from its actual value
Playback	Change signal value to a recorded value
Suppress	Prevent an ECU sending messages
Flooding	Inject selected IDs with high frequency

7.3 CAN Payload Data-based Intrusion Detection

This section provides a comprehensive overview of the proposed novel AE-based model and its anomaly detection procedure.

7.3.1 Datasets

In this study, we utilise two publicly available datasets: the ROAD CAN intrusion dataset [30] and SynCAN [70], to evaluate the proposed model. Correlated signal, max speedometer, reverse light on and off, and their masquerade versions from the ROAD dataset were selected for performance evaluation. The SynCAN dataset includes five types of advanced attacks conducted during post-processing, as summarized in Table 7.1. Despite the synthetic nature of the SynCAN dataset and its attacks, it serves as a suitable dataset for evaluating payload-based IDSs, as each attack targets multiple signals during different time intervals. The HCRL datasets, however, are not utilised for evaluation due to their limitations in assessing payload-based IDSs. One such limitation is the unavailability of a large benign dataset for model training.

7.3.2 Data Pre-processing

CAN data, characterized by the transmission of only one frame at a given time due to the priority-based arbitration mechanism, can be viewed as time series data. Previous research [98, 70, 101] has leveraged the sequential nature of CAN frames, employing deep learning models capable of processing sequential data, such as LSTM and GRU. While these RNN-based models exhibit high detection rates, there are certain drawbacks when applied to CAN intrusion detection. Typically, LSTM or GRU nodes have a higher number of trainable parameters, making them computationally more expensive than feed-forward neural networks with an equivalent number of nodes. This poses a significant challenge considering the limited computational resources available in IVNs. Moreover, research has highlighted the importance of considering the payload data of other IDs for effective CAN intrusion detection [102]. In such cases, the input sequence needs to

be sufficiently large to capture vital associations from other IDs. However, this also raises the computational complexity of the model, and the input frame may encompass numerous unassociated variables, potentially leading to overfitting. This will lead to reduce the detection capability of the model in a real-world deployment.

To address these problems, we transform CAN payload data into a format suitable for anomaly detection. First, the hexadecimal 64-bit CAN payload is segmented into eight bytes, which are subsequently converted into integer values. These eight features encompass a mix of constant or empty, categorical, and numerical discrete variables, with values ranging from 0 to 255. Normalization is applied to all features, ensuring they lie within the range of zero to one through ID feature-wise min-max scaling. This normalization aids in preventing slow and unstable training, as well as mitigating the issue of exploding gradients. A one-dimensional array is employed to store the most recent values of other CAN IDs alongside the current ID. This preserves the sequential nature of the CAN data and facilitates the learning of the context of payload values. Notably, this approach diverges from the one used in [66], which considered a few pre-identified variables and created 2-D frames for a specific time window. In contrast, the proposed data structure in this work considers context from all other IDs, as time-based window selection may overlook associations from certain IDs. Additionally, the proposed approach assumes no prior knowledge of CAN specifications, which are often not available for open access. Table 7.2 shows a subset of CAN IDs and features to illustrate the data transformation, while Table 7.2a shows CAN transmission between 77.04383s and 77.04387s for four CAN IDs. D1 and D2 denote the first and second features of the CAN payload, with each ID having up to eight features (D1...D8). The transformed data snapshot is presented in Table 7.2b. In this representation, each array (row) undergoes an update with the current ID's payload values. For instance, when ID 125 transmits at 77.04387s, the array update occurs for features 125_D1 and 125_D2, while the other features in the array retain their previous values. These features serve as contextual information for CAN ID 125, encapsulating the most recently transmitted values on the CAN bus. The array holds the latest values for all features of the current ID and associated variables for that ID

The ROAD dataset utilises zero padding to populate empty variables, resulting in a 64-bit payload field. Within the transformed data structure, variables that consistently hold a value of zero throughout the dataset are assumed instances where zero padding was applied. As a result, these variables are omitted from the data structure. Conversely, the SynCAN dataset does not necessitate any preprocessing, as variable values are already normalized, and empty variables are absent. However, for realistic CAN data resembling

the format of the ROAD dataset, it is crucial to implement these preprocessing steps. The modified data structure facilitates the implementation of a single model to detect anomalies across all CAN IDs, eliminating the need to implement separate models for each CAN ID.

Time	CAN ID	D1	D2
77.04383	125	0.1142	0.0000
77.04384	354	1.0000	0.4481
77.04385	5E1	0.1574	1.0000
77.04386	0A7	0.3333	0.8470
77.04387	125	0.1152	0.3278

(a) Snapshot of the normalized CAN payload. This represents only four CAN IDs and two features (D1, D2) out of eight features

Time	CAN ID	125_D1	125_D2	354_D1	354_D2	5E1_D1	5E1_D2	0A7_D1	0A7_D2
77.04383	125	**0.1142**	**0.0000**	0.2000	0.4481	0.1574	1.0000	0.1759	0.9803
77.04384	354	0.1142	0.0000	**1.0000**	**0.4481**	0.1574	1.0000	0.1759	0.9803
77.04385	5E1	0.1142	0.0000	1.0000	0.4481	**0.1574**	**1.0000**	0.1759	0.9803
77.04386	0A7	0.1142	0.0000	1.0000	0.4481	0.1574	1.0000	**0.3333**	**0.8470**
77.04387	125	**0.1152**	**0.3278**	1.0000	0.4481	0.1574	1.0000	0.3333	0.8470

(b) Snapshot of the transformed CAN payload. Each row represents the change in the variables over time as each CAN ID transmits. Variable values of the current ID are shown in bold.

Table 7.2: Data transformation from normalized CAN payload to amalgamated CAN payload. Only a subset of IDs and features are shown

7.3.3 Feature Selection

The transformed data structure outlined in Table 7.2 may encompass up to 8N features, where N represents the total number of IDs. Incorporating all these features could significantly augment the complexity and computational overhead of the IDS. An alternative strategy involves prioritizing essential features, which can help mitigate complexity and offer a practical solution with near real-time detection capabilities. However, discerning the significance of features for each ID presents a challenge in the absence of CAN specification knowledge. One plausible approach is to leverage feature associations to identify associated feature combinations. Prior research [97, 214, 215] has utilised the Pearson correlation coefficient to discern clusters of important variables, while others, such as [195, 66], have employed feature correlation to detect anomalies. In [97], the authors utilised raw payload values as features, while others focused on decoded signal values.

Based on the analysis outlined in Section 7.4.1, it is apparent that many payload features exhibit a limited number of unique categorical values. For nominal categorical features,

such as the 3rd byte of ID 0D0, employing the Pearson correlation may not be suitable for estimating feature associations. Consequently, relying solely on Pearson correlation can impede the identification of associated features in CAN payload data. For example, in [215], one of the selected sensor values was the Gear. This particular sensor exhibited the lowest Pearson correlation with other sensor values. Remarkably, the Gear sensor had only 7 possible values, akin to a nominal categorical variable. Consequently, the Pearson correlation method failed to capture the highly associated variables in this specific instance.

Cramér's V statistic, based on Pearson's chi-squared statistic, quantifies the strength of association between two discrete variables with two or more levels [216]. It is calculated as follows:

$$V = \sqrt{\frac{\varphi^2}{min(k-1, r-1)}} = \sqrt{\frac{\chi^2/n}{min(k-1, r-1)}} \tag{7.1}$$

In this equation, φ represents the phi coefficient, χ^2 denotes the chi-square statistic, n stands for the total number of observations, k represents the number of columns, and r denotes the number of rows in the contingency table. However, for finite samples, Cramér's V can exhibit significant bias. To address this concern, a bias correction method was introduced by [217]. The corrected value is expressed as follows:

$$\hat{V} = \sqrt{\frac{\hat{\varphi}^2}{min(\hat{k}-1, \hat{r}-1)}} \tag{7.2}$$

where,

$$\hat{\varphi}^2 = max\left(0, \varphi^2 - \frac{(k-1)(r-1)}{n-1}\right)$$

and

$$\hat{k} = k - \frac{(k-1)^2}{n-1}, \hat{r} = r - \frac{(r-1)^2}{n-1}$$

Cramér's V statistic ranges from 0 to 1, with 0 indicating no association between variables and 1 indicating a perfect association between variables [218]. This work utilises the corrected Cramér's $\hat{V}$ statistic to identify the associated features for all IDs. The associated feature selection procedure is shown in Algorithm 5. First, this algorithm selects the features X_1 of an ID and calculates the strength of associations $\hat{V}$

with the feature X_2 of other IDs using the contingency tables. The objective of this is to remove unassociated features from the other IDs. Therefore, association calculation between features of the same ID is not necessary as all features of the ID are kept in the array without removing them. Threshold λ is used to control the number of feature selections based on the desired strength of associations. If a feature does not have highly associated features with over λ, then the feature with the highest $\hat{V}$ is selected as the associated feature for the particular feature. This ensures that the model considers all features in the dataset with at least one associated feature. Based on this algorithm, unassociated features of the transformed CAN payload (see Table 7.2) are removed using zero padding. This converts the dense array to a sparse array. As a result of this feature selection approach, it reduces the dataset size, which requires learning the benign variable pattern compared to having all associated and non-associated features. In a production environment, previous frame values of the transformed CAN payload and the list of associated variables of an ID could be used to update the latest array. Therefore, it only requires storing the latest dense frames and associated feature dictionaries in memory.

7.3.4 Latent AE-Improved Autoencoder Architecture

Anomaly detection based on AEs relies on the assumption that benign data exhibit smaller reconstruction errors due to learned patterns, while anomalous data display larger reconstruction errors. In vanilla AEs, this reconstruction error acts as the anomaly score, distinguishing between benign and anomalous samples. However, this assumption may not consistently hold in practical situations. There are instances where AEs generalize so well that they can effectively reconstruct anomalies, leading to a reconstruction error that is not sufficiently large to flag them as anomalies. Consequently, this can result in a significant number of false negatives.

This work addresses the issue of overgeneralization of vanilla AE by introducing an additional small AE model within the latent space. While previous research [45, 211, 212] leveraged the latent space to enhance anomaly detection, none of these studies integrated an extra AE within the latent space, distinguishing our model. The proposed model, Latent AE, comprises three components: an encoder, a decoder, and a latent space AE. As depicted in Figure 7.1, the black line illustrates the training process, while the blue line represents the inference process. Initially, given an input X, the encoder derives the latent space z. The decoder utilises this latent space to reconstruct the input. Simultaneously,

Algorithm 5 Associated feature selection procedure

Input: CAN ID list L, Features X, Threshold λ
Output: Associated feature dictionary D, Unassociated feature dictionary D'

```
Init: D = { }
for id ∈ L do
    Init: Feature list F = [ ]
    Select X_1 = [x_1, ..., x_8] ⊂ X, X_1 ∈ id
    Select X_2 = [x_9, ..., x_n] ⊂ X, X_2 ∉ id
    for i ∈ X_1 do
        Init: Highest associated feature X_h = 0
        for j ∈ X_2 do
            Create contingency table T
            Compute n, k, r, χ^2                          ▷ Using T
            Compute V̂
            Update X_h
            if V̂ > λ then
                F.append[j]
            end if
        end for
        if len[F] = 0 then
            F.append[X_h]
        end if
    end for
    D[id] = F
    D'[id] = F'                          ▷ F' is the complement of F
end for
```

another small AE is trained to reconstruct the acquired latent space z. Throughout the training, the parameters of the encoder, decoder, and latent space AE are adjusted to minimize reconstruction errors through backpropagation and gradient descent. Both AEs are trained to reconstruct benign samples with low reconstruction errors E_1 and E_2. During inference, indicated by the blue line, the encoded input z serves as an input to the latent space AE model for reconstructing the latent input. Reconstruction error E_2 tends to be small for benign samples and large for anomalous samples. If E_2 surpasses a predefined latent threshold μ, the reconstructed latent input $\hat{z}$ is used as input to the decoder g_θ. This enforces the reconstruction of the original input with a significant reconstruction error for anomalous samples. Conversely, if E_2 falls below the predefined latent threshold μ, the original encoded input z is used as input to the decoder g_θ.

The anomaly detection process of the Latent AE is shown in Algorithm 6. This procedure closely resembles the CAN-ODTL anomaly detection algorithm proposed in Chapter 6. First, it calculates the input reconstruction error E_1 using both AEs. Subsequently, it categorizes each frame in the observation window as a weak anomaly or benign based on a predefined anomaly threshold ω. The window status is then identified as benign or anomalous using a predefined window threshold ψ and the count of anomalous frames over the total number of frames.

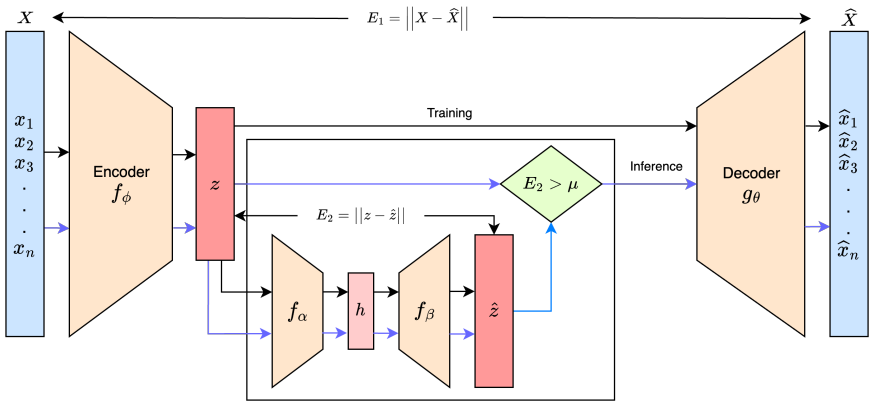


Figure 7.1: Overview of the Latent AE. The black line indicates the training process while the blue line indicates the inference process

7.3.5 Threshold Estimation

Latent AE requires to have three thresholds. Similar to the threshold estimation in the GRU-based model, a distinct benign dataset is utilised to estimate these thresholds. The

Algorithm 6 Latent AE anomaly detection

Input: Streaming CAN data F, Latent threshold μ, Anomaly threshold ω, Window threshold ψ, Time window T
Output: Anomaly status for each window

while F is not empty **do**
  read x, time_stamp t, t_min
  while $t - t\_min \leq T$ **do**
    Init: Benign count $C_b = 0$, Anomaly count $C_a = 0$
    $z = f_\phi(x)$
    $\hat{z} = f_\beta(f_\alpha(z))$
    Compute $E_2 = ||z - \hat{z}||$
    if $E_2 > \mu$ **then**
      $z \leftarrow \hat{z}$
    end if
    $\hat{x} = g_\theta(z)$
    Compute $E_1 = ||x - \hat{x}||$
    if $E_1 > \omega$ **then**
      Declare x as a weak anomaly
      $C_a = C_a + 1$
    else
      Declare x as a benign
      $C_b = C_b + 1$
    end if
  end while
  if $C_a/(C_a + C_b) > \psi$ **then**
    Return Anomaly
  else
    Return Benign
  end if
  $t\_min \leftarrow t$
end while

latent threshold μ serves to distinguish between benign and anomalous frames in the latent space. This threshold (μ) can be estimated by considering the highest reconstruction errors E_2 for the chosen benign dataset. Similarly, the anomaly threshold ω can be estimated considering the highest reconstruction errors E_1 for the input benign data. Since payload values depend on the associated IDs, both the latent and anomaly thresholds are estimated for each CAN ID. This approach helps in identifying anomalies specific to each ID, rather than applying a common threshold for all IDs. Although the highest reconstruction errors are ideal as threshold values to minimize false positives, it is important to consider that there might be a few benign frames in the threshold estimation benign dataset that were not observed during the training phase. Consequently, these benign frames may exhibit higher reconstruction errors. Therefore, we determine the N^{th} highest quantile values as the anomaly thresholds μ, ω, allowing for a very small percentage to account for benign anomalies. The window threshold ψ is defined in a manner that minimizes the false positive rate for a fixed window size of time T. A summary of all thresholds employed in the payload-based IDS is provided in Table 7.3

Table 7.3: Summary of thresholds used in CAN payload-based IDS

Notation	**Meaning**	**Description**
λ	Feature selection	This is used to control the number of associated feature selections. Start with a higher association threshold, such as 0.95, and gradually reduce it until the majority of features have at least one highly associated feature.
μ	Latent threshold	This is used to distinguish benign and anomalous frames in the latent space. The latent space AE is utilised to compute the reconstruction errors for all IDs in a benign dataset. The N^{th} highest quantile values are then calculated for each CAN ID.
ω	Anomaly threshold	The reconstruction errors for all IDs are calculated using the Latent AE for a benign dataset. Subsequently, the N^{th} highest quantile values are calculated for each CAN ID.
ψ	Window threshold	The trained Latent AE is used to determine the anomalous or benign status of each frame in a benign dataset using the calculated ω. The average false positive rate is then computed for each time window T, and this value is set as the threshold

CAN payload-based IDSs can effectively detect both injections and masquerade attacks as both might change the payload field patterns. However, due to the complexity of the payload field compared to the ID field, it requires selecting important payload features to achieve near real-time detection in a resource-constrained environment. The challenge lies in feature selection, which may overlook some significant features due to the lack of knowledge regarding CAN data specifications.

7.3.6 Ensemble IDS

Ensemble models in machine learning combine predictions from multiple models to enhance overall performance. Consequently, an ensemble IDS that incorporates both CAN ID and payload-based models can effectively address the limitations of individual models. For instance, in cases where injection attacks subtly alter payload values, resulting in minimal reconstruction errors, the Latent AE model might struggle to detect such changes. However, the GRU-based model could still identify these attacks through changes in the ID sequence. Conversely, sophisticated masquerade attacks that maintain unchanged ID sequences may evade detection by the GRU-based model but could be identified by the Latent AE model. In contrast, Latent AE has the capability to detect these attacks. Therefore, an ensemble combining both models enhances overall attack detection capabilities. While the time-based model which use in Chapter 5 could also be integrated into the ensemble, this chapter specifically focuses on the integration of AI-based GRU and AE models for the ensemble IDS.

The proposed CAN ID-based IDS and payload-based IDS classify benign and anomalous windows in streaming CAN data. The ensemble prediction is derived by utilising the predictions from both algorithms, as demonstrated in Algorithm 7. Algorithm 3, providing predictions for the GRU-based model, and Algorithm 6, yielding predictions for the AE-based model, operate simultaneously. The output statuses from both algorithms are combined using an OR operator to determine the final prediction for the ensemble IDS.

Algorithm 7 Ensemble IDS anomaly detection

Input: Streaming CAN data F
Output: Anomaly status for each window
1: **while** F is not empty **do**
2: $output\_1 \leftarrow$ *Algorithm 3* ▷ Prediction of the GRU-based model
3: $output\_2 \leftarrow$ *Algorithm 6* ▷ Prediction of the AE-based model
4: **if** $output\_1$ *or* $output\_2$ = Anomaly **then**
5: Return Anomaly
6: **else**
7: Return Benign
8: **end if**
9: **end while**

7.4 Evaluation and Performance Results

This section presents the analysis of CAN bus data, identification of feature associations, specification of algorithm parameters, and performance evaluation.

7.4.1 CAN Payload Data Analysis

DBC files contain crucial details such as signal definitions, message transmission frequencies, and ECU information [182]. According to the specified definitions, the payload field may encompass sensor data, category data, constant data, or cyclical counter data [97]. Reverse engineering efforts on CAN payloads have also unveiled physical values, constants, and counter or CRC values [219]. In [29], these fields were categorized as constant, multi-value, and sensor values. Since the boundaries of these fields are unknown, the majority of previous payload-based IDS discussed in Section 3.3.2, involve converting the 64-bit CAN payload into 8 bytes and treating each byte as a variable (feature), with each feature value falling within the range of 0 to 255. IDs with a payload shorter than 64 bits have empty features. This study adopts the same conversion technique and analyzes the CAN payload using the ROAD CAN intrusion dataset [30] to understand benign traffic patterns.

The ROAD dataset comprises 106 CAN IDs, resulting in 848 features (106 x 8) when converting 64 bits from binary to decimal. This dataset employs zero padding for empty features. The analysis of payload field data involves using the combined dataset of all the benign datasets listed in Table 5.1. Figure 7.2 illustrates the distribution of unique values across these 848 features. Based on this distribution, 249 features (29%) are identified as either constants or empty, while 321 features (37%) exhibit unique values ranging from 1 to 9. In contrast, 40 features (0.08%) have 256 unique values, representing features that encompass every discrete value between 0 to 255. Features that are neither constant nor empty can be categorized as either nominal or ordinal categorical features. For instance, the 3rd byte of ID 0D0 communicates the reverse light status, with only two possible values (4 and 12) indicating the reverse light on and off status. This makes it a nominal categorical feature. On the other hand, the 6th byte of 0D0 communicates the speedometer signal, which can assume any value between 0 and 255. This type of feature may exhibit a clear ordering of categories, making it an ordinal categorical feature. Similar constant, ordinal, and nominal variable patterns are observed in other features. However, without access to the DBC file or the associated information, accurately distinguishing between nominal and ordinal status for categorical features

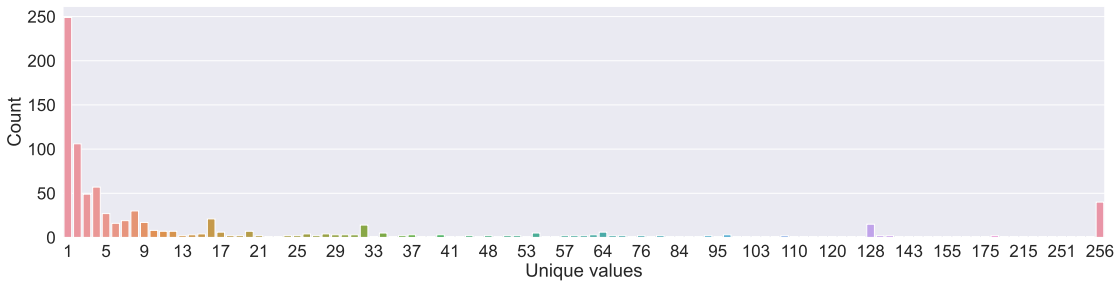


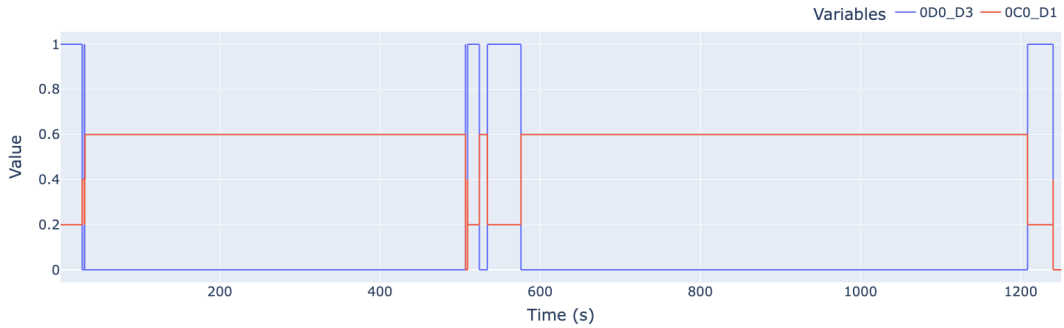
Figure 7.2: Unique value distribution for ROAD dataset features

remains challenging. As attackers could potentially target any feature, the ability to detect attacks on various feature types becomes a crucial aspect of a CAN payload-based IDS.

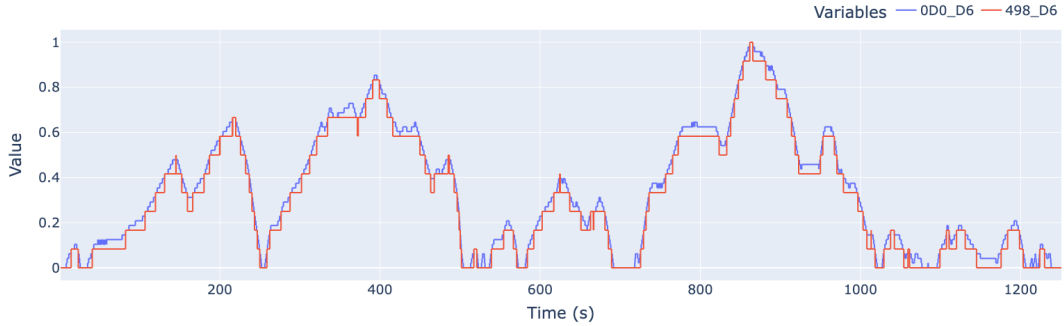
7.4.2 Feature Association

Selecting the appropriate threshold (λ) is critical to identify highly associated features for a particular feature. Small λ values result in a complex model by selecting too many features, while larger λ values choose only highly associated features, making the model less complex. It is essential to identify at least one highly associated feature for each payload variable, as having more associated variables enhances the IDS's detection capability against attacks on that variable. This is because the presence of additional associated variables can disrupt the multiple expected associations and result in higher reconstruction errors. In determining the appropriate threshold (λ), we started with 0.95 and successively decreased it by 0.05. At $\lambda = 0.8$, it successfully identified at least one associated variable for 98% of the variables. For the remaining variables, we selected the highest associated variable below the 0.8 threshold. The majority of variables had two or more associated variables. While further decreasing the threshold could reveal more associated variables, this would escalate the computational cost of the IDS. Therefore, we opted to strike a balance between capturing essential associations and maintaining computational efficiency. Hence, we set λ to 0.8 for the ROAD dataset. Considering that the SynCAN dataset comprises only 20 signals, we adjust λ to 0.5 using the same methodology to capture a greater number of associated variables. In the ROAD dataset, attacks involving the reverse light on and off target a single bit in the third byte of ID 0D0. As mentioned in Section 7.4.1, feature 0D0_D3 can be considered as a nominal categorical feature as it has only 2 values.

Algorithm 5 identifies four features that exhibit a high level of association with feature



(a) Association between 0D0_D3 and 0C0_D1



(b) Association between 0D0_D6 and 498_D6

Figure 7.3: ROAD dataset variable associations. The x-axis is the time, and y-axis is the normalized variable value

0D0_D3, with a correlation coefficient ($\hat{V}$) exceeding 0.99. Specifically, feature 0D0_D3 demonstrates a strong association of 0.998 with feature 0C0_D1, which has four distinct values. The value change of feature 0C0_D1 with respect to 0D0_D3 is depicted in Figure 7.3a, clearly illustrating the perfect association between these two nominal categorical features. In contrast, the Pearson correlation coefficient between 0C0_D1 and 0D0_D3 is moderate at 0.59. Furthermore, feature 0D0_D3 exhibits a strong association of 0.999 with feature 274_D7, which has 16 unique values. The Pearson correlation coefficient between these two variables is -0.35. Similar patterns can be observed for other nominal categorical variables. On the other hand, feature 0D0_D6 displays ordinal categorical behaviour with 63 distinct values, demonstrating the highest association (0.998) with feature 498_D6. Figure 7.3b visualizes the association between these two features, while the Pearson correlation coefficient between them is 0.996.

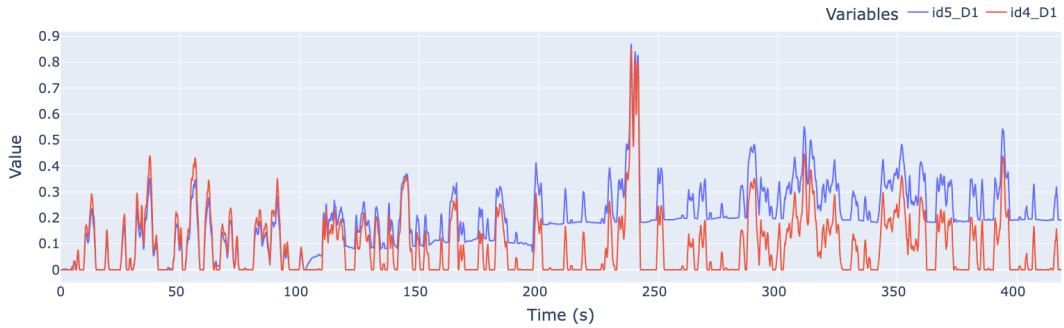


Figure 7.4: Association between id5_D1 and id4_D1 in SynCAN dataset. The x-axis is the time, and the y-axis is the normalized variable value

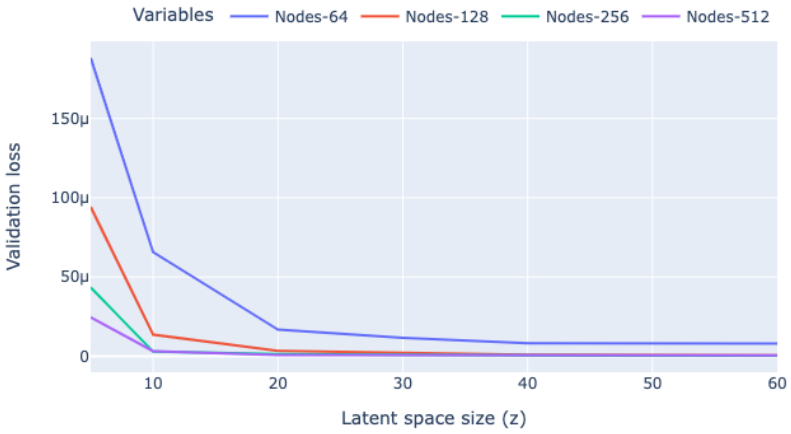
Similarly, the corrected Cramér's statistic highlights associated features within the SynCAN dataset. In particular, Figure 7.4 illustrates a notable association of 0.876 $\hat{V}$ between id5_D1 and id4_D1 features. Comparable associations are evident across all features in both datasets. These findings underscore the capability of $\hat{V}$ to discern both ordinal and nominal categorical associations, rendering it more suitable than Pearson correlation for identifying associated features in the CAN payload. During training, AEs can learn feature associations and subsequently identify abnormal associations during testing. Without knowledge of the exact CAN data specifications, it may be difficult for an attacker to manipulate all highly associated features to maintain the same level of associations during an attack.

7.4.3 Experimental Setup

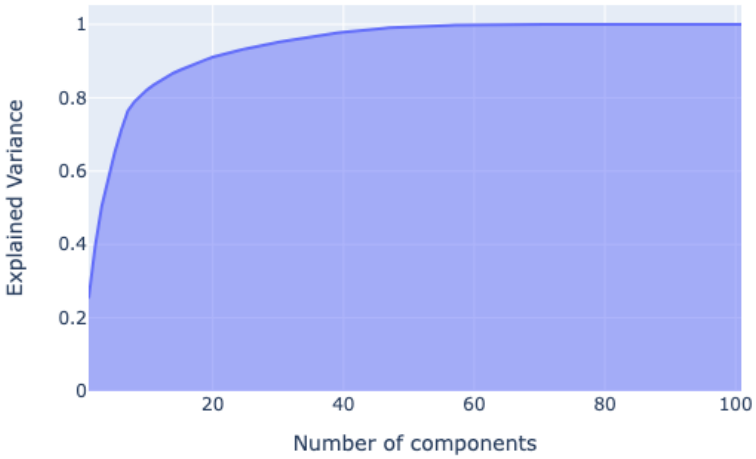
ROAD dataset

The training and threshold estimation datasets for the ROAD dataset were selected following a similar approach to the dataset selection of the CAN-CID IDS, as discussed in Section 5.4.2. A symmetric AE architecture is employed for both AEs. The transformed data structure includes 655 variables after removing the empty variables. To maintain a lightweight model, the encoder is constrained to have only two hidden layers, including the latent layer. A grid search is conducted to determine the optimal nodes for the hidden layers. The parameter space for the first hidden layer encompasses 64, 128, 256, and 512 nodes, while the latent space parameters range from 5 to 60 nodes, including increments of 5.

In Figure 7.5a, the validation loss, representing the reconstruction error for the validation



(a) Validation loss for different latent sizes



(b) Explained variances of input data for different latent space sizes

Figure 7.5: Latent space size selection

dataset, is illustrated across various latent sizes for different numbers of hidden nodes in the first hidden layer. The validation loss experiences a sharp decline up to 10 latent sizes across all nodes in the first layer. Following this, it marginally decreases up to 50 latent sizes, contingent upon the number of nodes in the first layer. Generally, a latent size ranging from 10 to 50 corresponds to the lowest validation loss. However, it is important to note that a higher number of nodes in an AE may result in overgeneralization, potentially compromising its efficacy for anomaly detection tasks. Mere minimization of the validation loss does not ensure optimal anomaly detection performance. Conversely, an overly simplistic AE structure might struggle to capture the data's variability adequately, thereby lacking the robustness to accurately reconstruct the inputs. Therefore, it is crucial to carefully choose the number of nodes in an AE. To aid in determining the optimal latent size for anomaly detection, PCA can offer valuable insights due to certain similarities between AEs and PCA [220]. PCA aims to identify orthogonal axes aligning with the data's greatest variability directions [220]. By analyzing the PCA variance explained graph, we can identify the number of components explaining varying levels of variability in the dataset. The latent space size of the AE, ranging from 10 to 50, corresponds to the number of principal components (PCs) in PCA that elucidate the input data variability, spanning from 80% to 99%. In other words, employing 10 to 50 principal components in PCA can explain 80% to 99% of the input data variability. A latent size of 10 explains 80% of the variability, while latent sizes of 20 and 47 explain 90% and 99% variability, respectively. This relationship is depicted in Figure 7.5b. Considering that only 10 principal components account for 80% of the data variability, opting for a latent size of 10 in the AE may not suffice, potentially failing to capture the intricacies of the data. This inadequacy is evident in Figure 7.5a, where a substantial number of nodes in the first layer is necessary for a latent size of 10 to attain a minimal reconstruction loss. In contrast, when employing 20 principal components, which explain 90% of the variability, incorporating 20 nodes in the latent space of the AE necessitates only 128 nodes in the first layer to attain a low validation loss. Hence, considering the model complexity, we opt for 128 nodes in the first layer and 20 nodes in the latent space. For the latent space AE, we adopt a shallow network comprising only one hidden layer. Given its shallow nature, the latent size is set to encompass 99% of the PC variability size, ensuring a satisfactory reconstruction of latent input data. Consequently, the latent space AE is configured with 18 nodes in the latent space, with 20 as the input dimension. The latent space AE is notably smaller in comparison to the vanilla AE, primarily due to the reduced input size. Given the frame transmission rate of approximately 2000 frames per second, attack datasets are divided into 25-millisecond windows to identify attack

occurrences, presenting a smaller window suitable for near real-time prediction. The window threshold is set to 0.03 based on the lowest false positive rate (average) for the benign dataset. Additionally, latent and anomaly thresholds are computed for each CAN ID, taking into account the reconstruction errors for the input frames. To accommodate a small margin for unseen benign data, the thresholds are determined using the 99.9th quantile values.

SynCAN dataset

The same methodology is applied to determine the parameters for the SynCAN dataset. Consequently, 32 and 15 nodes are chosen for the vanilla AE, with an input size of 20. The parameter space for the first hidden layer encompasses 8, 32, 64, and 128 nodes, whereas the latent space parameters include 5, 10, 15, and 20 nodes. In the case of the latent space AE, 11 nodes are designated for the latent layer. Unlike the real ROAD dataset, the SynCAN dataset comprises only 10 CAN IDs with lower transmit rates. Therefore, 100-millisecond windows are adopted, with a 0.02 window threshold. This results in approximately 100 CAN IDs per window, requiring the detection of at least two anomalous frames to classify the window as anomalous. For model training, three datasets were combined and utilised as the training datasets, while a distinct dataset was reserved for threshold estimation.

For the performance evaluation, both datasets categorize a window as anomalous (ground truth) if at least one frame within it is deemed anomalous. All AEs undergo training for 100 epochs, employing a batch size of 128. To prevent overfitting, early stopping mechanisms are employed. The learning rate is set to 0.0001 with the Adam optimizer. Relu activation functions are employed for all layers except the final layers. For computing reconstruction errors E_1 and E_2, MAE is utilised due to its robustness against outliers. The proposed algorithm is implemented using Python 3.8 with Tensorflow and Keras libraries. KerasTuner is utilised for the grid search process. All experiments are conducted on a MacBook M1 Pro equipped with 16 GB of RAM.

7.4.4 Results and Discussion

We compare the proposed model Latent AE with vanilla AE and two variants of the Latent AE: Latent AE-ND (Non-Decoder) and Latent AE-NT (Non-Threshold). Latent AE-ND solely utilises the encoder of the vanilla AE and the latent space AE, where the reconstruction error E_2 from the latent space AE is employed to identify anomalies. Conversely, Latent AE-NT eliminates the latent threshold μ and sends the reconstructed

latent space $\hat{z}$ into the decoder of the vanilla AE. These two models are employed to evaluate the effectiveness of the latent space AE. Additionally, OCSVM and a RNN-based model are used for comparison with Latent AE. In this regard, we adopt INDRA [101] as the RNN-based model, which shares a similar architecture with the model proposed in [102]. INDRA utilises a GRU network to reconstruct input frames of size 20, and this work adopts their network architecture to train one model for each CAN ID. For the ROAD dataset, the optimized parameter sequence for the input frame comprises 30 messages. The optimized hyperparameters for the OCSVM model include gamma values of 0.0001 for the SynCAN dataset and 0.01 for the ROAD dataset, along with nu values of 0.1 for SynCAN and 0.001 for ROAD. Model performance is evaluated using macro-averaged F1-score (F1), TP, TN, FP, and FN rates. The evaluation metrics in this approach are computed based on observation windows, as outlined in Algorithm 6. Specifically, the counts of benign and anomaly instances within these observation windows are utilised to derive the evaluation metrics. Given that our training dataset exclusively comprises benign data and the test datasets consist of attack data, employing cross-validation is not feasible in this scenario. Instead, we conduct multiple independent experiments, averaging the results from 10 different realizations, to ensure unbiased and reliable performance evaluation.

ROAD Dataset Attack Detection

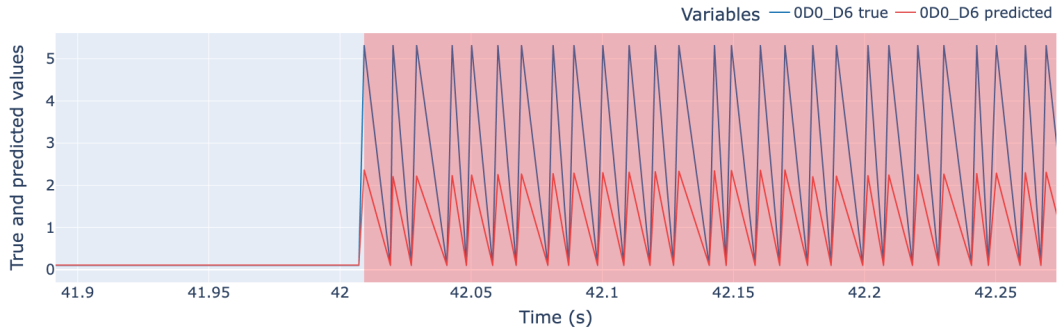
Table 7.4 presents the detection outcomes of Latent AE and its variants in comparison to OCSVM and the baseline model INDRA. In the correlated signal attack, ID 6E0, responsible for transmitting the speeds of all four wheels, is targeted. This attack manipulates all payload values to malicious ones, thereby generating both point and contextual anomalies that can be readily detected by learning the benign ranges. As anticipated, all models exhibit a higher detection rate for this attack. Latent AE and its variants achieve a 100% detection rate (TP). Similarly, the masquerade version of the attack, which only removes benign samples of the targeted ID during the attack, also attains the same level of detection. However, OCSVM demonstrates a higher false positive rate for both attacks. OCSVM's sensitivity to gamma and nu parameters allows for the adjustment of decision boundaries, but we observe higher FN rates and consequently lower F1-scores for other gamma and nu values. The large dataset size and its high dimensionality might contribute to the comparatively lower performance of the OCSVM model.

The Max speedometer attack involves changing the 6th byte of ID 0D0 to its maximum value (255), representing a nominal categorical variable. The GRU-based INDRA

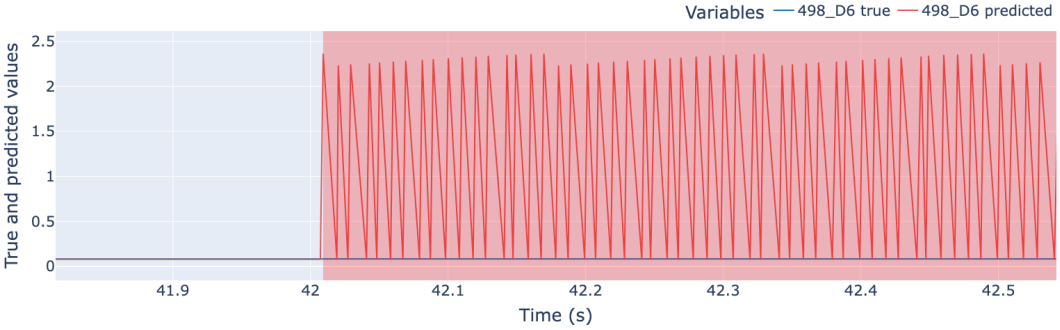
Table 7.4: Comparison of Latent AE, Latent AE variants and baseline models detection performance of ROAD dataset

Attack	Model	F1	TP	TN	FP	FN
Correlated signal	OCSVM	89.3%	100%	67.9%	32.1%	0.0%
	INDRA	99.3%	100%	98.8%	1.2%	0.0%
	Vanilla AE	**100**%	100%	100%	0.0%	0.0%
	Latent AE-ND	**100**%	100%	100%	0.0%	0.0%
	Latent AE-NT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	**100**%	100%	100%	0.0%	0.0%
Correlated signal masquerades	OCSVM	89.3%	100%	67.9%	32.1%	0.0%
	INDRA	99.3%	100%	98.8%	1.2%	0.0%
	Vanilla AE	**100**%	100%	100%	0.0%	0.0%
	Latent AE-ND	**100**%	100%	100%	0.0%	0.0%
	Latent AE-NT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	**100**%	100%	100%	0.0%	0.0%
Max speedometer	OCSVM	93.5%	100%	89.6%	10.3%	0.0%
	INDRA	99.6%	100%	99.3%	0.7%	0.0%
	Vanilla AE	**100**%	100%	100%	0.0%	0.0%
	Latent AE-ND	**100**%	100%	100%	0.0%	0.0%
	Latent AE-NT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	**100**%	100%	100%	0.0%	0.0%
Max speedometer masquerades	OCSVM	93.5%	100%	89.6%	10.3%	0.0%
	INDRA	99.6%	100%	99.3%	0.7%	0.0%
	Vanilla AE	**100**%	100%	100%	0.0%	0.0%
	Latent AE-ND	**100**%	100%	100%	0.0%	0.0%
	Latent AE-NT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	**100**%	100%	100%	0.0%	0.0%
Reverse light on	OCSVM	33.1%	0.0%	96.2%	3.7%	100%
	INDRA	33.8%	0.0%	99.1%	0.9%	100%
	Vanilla AE	34.0%	0.0%	100%	0.0%	100%
	Latent AE-ND	**100**%	100%	100%	0.0%	0.0%
	Latent AE-NT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	**100**%	100%	100%	0.0%	0.0%
Reverse light on masquerade	OCSVM	33.1%	0.0%	96.2%	3.7%	100%
	INDRA	33.8%	0.0%	99.1%	0.9%	100%
	Vanilla AE	34.0%	0.0%	100%	0.0%	100%
	Latent AE-ND	**100**%	100%	100%	0.0%	0.0%
	Latent AE-NT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	**100**%	100%	100%	0.0%	0.0%
Reverse light off	OCSVM	38.1%	0.0%	97.1%	2.9%	100%
	INDRA	40.2%	0.0%	99.7%	0.3%	100%
	Vanilla AE	36.0%	0.0%	100%	0.0%	100%
	Latent AE-ND	**100**%	100%	100%	0.0%	0.0%
	Latent AE-NT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	**100**%	100%	100%	0.0%	0.0%
Reverse light off masquerade	OCSVM	38.1%	0.0%	97.1%	2.9%	100%
	INDRA	40.2%	0.0%	99.7%	0.3%	100%
	Vanilla AE	36.0%	0.0%	100%	0.0%	100%
	Latent AE-ND	**100**%	100%	100%	0.0%	0.0%
	Latent AE-NT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	**100**%	100%	100%	0.0%	0.0%

model can detect the sudden significant value increment as anomalous by learning signal patterns and using signal level thresholds. OCSVM also demonstrates better detection performance compared to correlated signal attack detection. In contrast, Latent AE and its variants can detect this attack from two perspectives: significant value change and



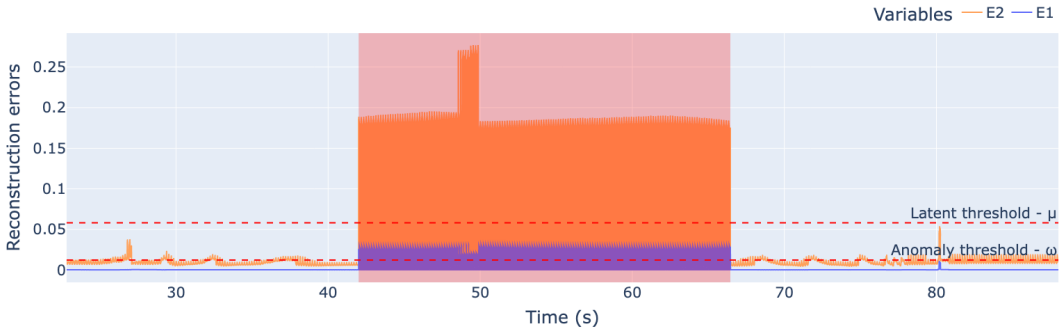
(a) 0D0_D6 true and predicted values



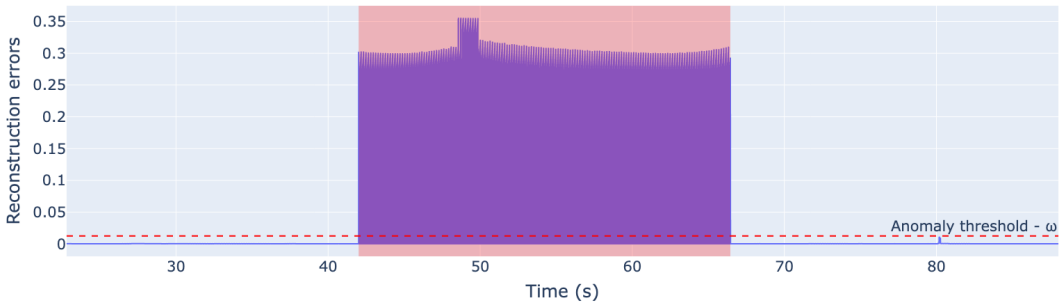
(b) 498_D6 true and predicted values

Figure 7.6: Max speedometer attack true and predicted values. Shaded area represents the attack period

alteration of feature associations. This can be illustrated using Figure 7.6, which displays a snapshot of the attack dataset over a few seconds. During the attack period, spikes represent the attack frames, while normal values represent benign frames. AEs fail to reconstruct the spikes with the same magnitude, as these large spikes were not present in the benign training data. Consequently, variable 0D0_D6 yields a large reconstruction error, as depicted in Figure 7.6a. On the other hand, 0D0_D6 exhibits a higher association with feature 498_D6. Due to this learned association, AE attempts to reconstruct feature 498_D6 in a manner that maintains the same level of association with feature 0D0_D6. However, since the true value of 498_D6 remains unchanged by the attack, this results in a significant reconstruction error. Since vanilla AE, Latent AE, and its variants consider ID-based massage level reconstruction error, collectively, these signals generate a significant reconstruction error, aiding in easy detection of this attack. Attack detection is illustrated in Figure 7.7. As depicted in Figure 7.7a, both vanilla AE and latent space AEs are capable of detecting this attack alone. Therefore, vanilla AE, Latent



(a) Reconstruction errors for vanilla and latent space AEs

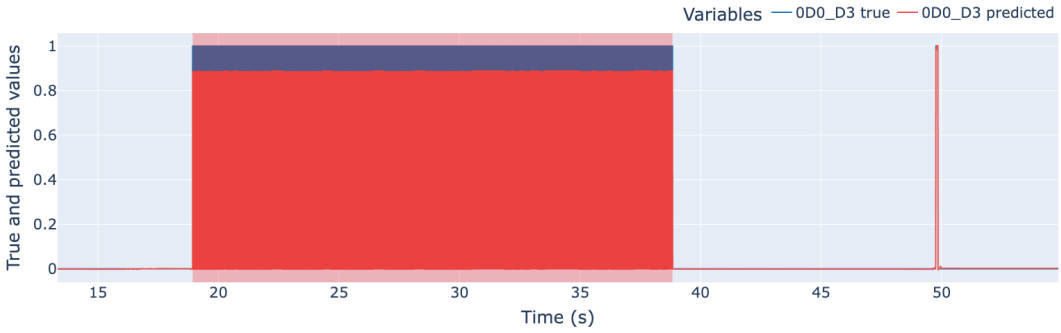


(b) Reconstruction errors for Latent AE

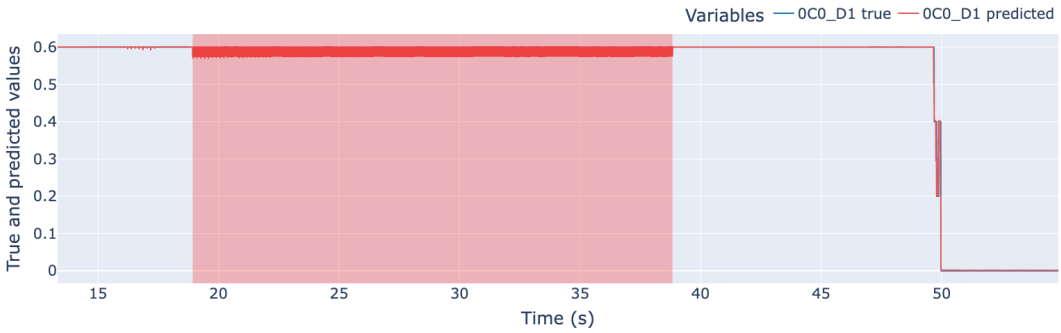
Figure 7.7: Max speedometer attack reconstruction errors. E_2 and E_1 represents latent and vanilla AE reconstruction errors respectively.

AE-ND, and Latent AE-NT achieve a 100% detection rate. Figure 7.7b demonstrates that Latent AE amplifies the anomaly reconstruction error, thereby enhancing the likelihood of attack detection. Similar performance can be observed for the masquerade attack as well.

The Reverse light on attack targets a single bit of the 3rd byte of ID 0D0, turning on the reverse light while the vehicle is in drive gear. Unlike other attacks described earlier, the reverse light on attack does not alter the feature value into an unseen value, as 0D0_D3 only takes two values. Consequently, detecting this attack relies on identifying discrepancies in feature associations. INDRA, with individual models for each CAN ID, does not leverage feature dependencies and therefore struggles to detect such intricate attacks. OCSVM also has limited effectiveness in detecting these attacks. Conversely, vanilla AE, which utilises our transformed data structure, should theoretically be capable of detecting the attack. However, it too fails to identify the reverse light on attack.



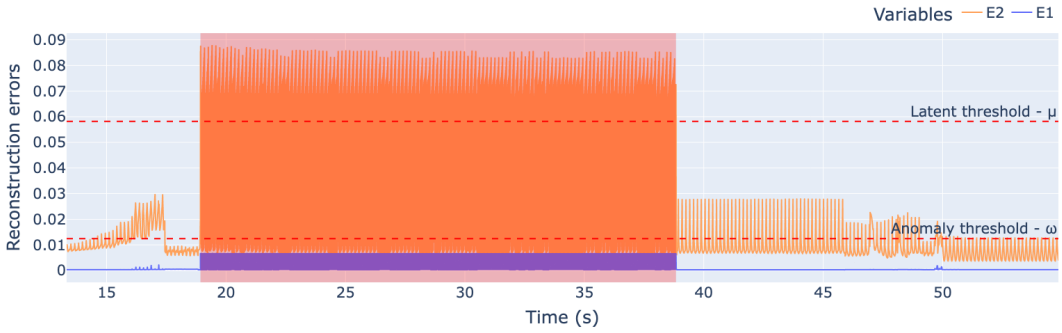
(a) 0D0_D3 true and predicted values



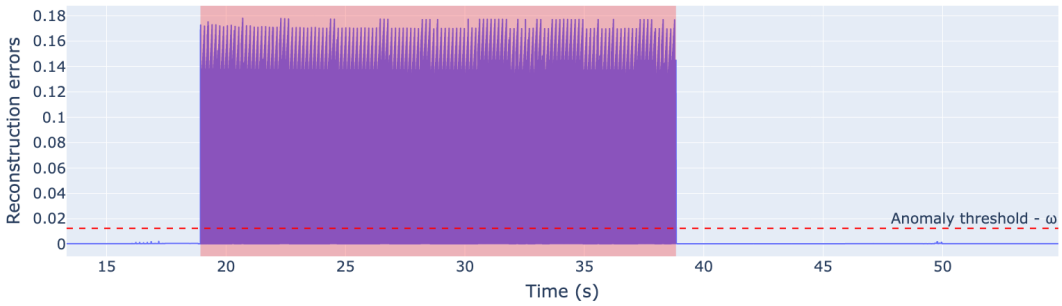
(b) 0C0_D1 true and predicted values

Figure 7.8: Reverse light on attack true and predicted values.

Figure 7.8 The reverse light on attack generates a minimal reconstruction error for the attack feature 0D0_D3 (Figure 7.8a) and its associated feature 0C0_D1 (Figure 7.8b). This pattern persists across all associated features of 0D0_D3. This phenomenon could stem from the well-known problem of overgeneralization in vanilla AE, reconstructing anomalous data alongside benign data. Despite our efforts to keep the model architecture simple by limiting the number of nodes in hidden layers for lightweight modeling, vanilla AE may still generalize too effectively for features with a narrow range of unique values (2-10). This behaviour persists across different model architectures, whether simple or complex. In contrast, Latent AE and its variants achieve a 100% detection rate for both reverse light on and masquerade attacks. Figure 7.9a illustrates the reconstruction errors for vanilla AE and latent space AEs. Vanilla AE yields a small reconstruction error insufficient for detecting anomalous messages. However, the latent space AE produces a significantly larger reconstruction error, effectively detecting the attacks. Latent AE further amplifies this reconstruction error due to the anomalous input to the decoder of vanilla AE (Figure 7.9b). This indicates that vanilla AE may be overgeneralized during



(a) Reconstruction errors for vanilla and latent space AEs



(b) Reconstruction errors for Latent AE

Figure 7.9: Reverse light on attack reconstruction errors

the decoding phase, but these attacks can still be identified in the latent space using the latent space AE. A restricted number of layers and nodes in the decoder of vanilla AE does not mitigate overgeneralization, as it also fails to reconstruct benign frames accurately. During attack periods, unassociated features reconstruct inputs similarly to benign periods, resulting in no false positives (FPs). Similar levels of detection are observed for the reverse light off and its masquerade attack version.

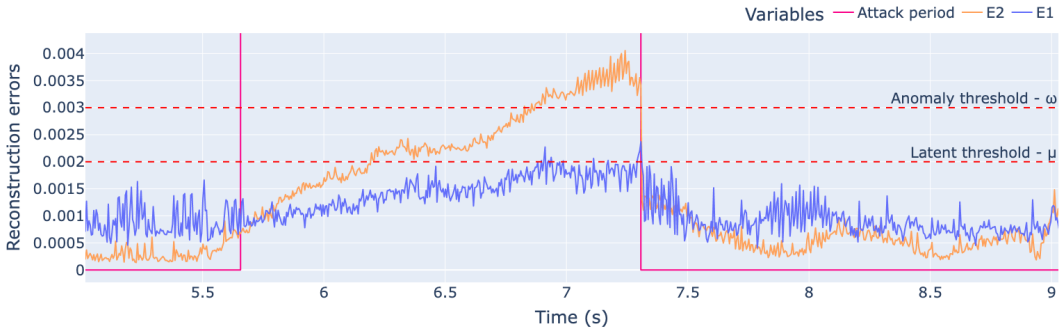
The analysis of CAN payload data in Section 7.4.1 reveals that most CAN features exhibit a restricted range of unique values (Figure 7.2). As demonstrated earlier, vanilla AE's inability to detect attacks on these features due to overgeneralization underscores the necessity for Latent AE in effectively identifying diverse CAN bus attacks. Furthermore, regardless of the specific alignment with the actual payload features, Latent AE achieved 100% detection rate across all attacks using the selected eight-byte features.

Table 7.5: Comparison of Latent AE, Latent AE variants and baseline models detection performance of SynCAN dataset

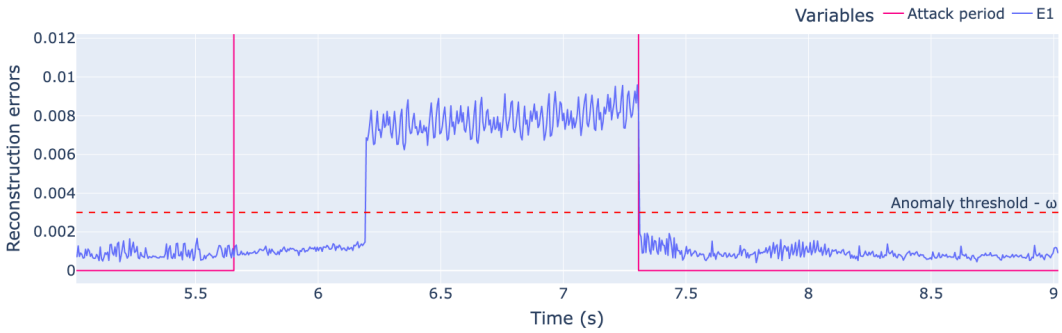
Attack	Model	F1	TP	TN	FP	FN
Plateau	OCSVM	57.0%	19.6%	92.4%	7.5%	80.3%
	INDRA	70.2%	39.7%	94.8%	5.1%	60.2%
	Vanilla AE	88.1%	70.5%	99.3%	0.7%	29.4%
	Latent AE-ND	88.1%	66.6%	99.7%	0.2%	33.3%
	Latent AE NT	84.3%	72.4%	94.8%	5.1%	27.5%
	Latent AE	**92.6**%	76.4%	99.3%	0.7%	23.5%
Continuous	OCSVM	53.7%	12.0%	93.6%	6.3%	87.9%
	INDRA	82.0%	56.2%	98.7%	1.2%	43.7%
	Vanilla AE	93.0%	81.7%	99.0%	0.9%	18.2%
	Latent AE-ND	91.4%	72.3%	99.8%	0.1%	27.6%
	Latent AE-NT	90.2%	85.8%	96.8%	3.2%	14.2%
	Latent AE	**95.4**%	84.1%	99.1%	0.8%	15.8%
Playback	OCSVM	49.8%	4.3%	97.0%	2.9%	95.6%
	INDRA	81.2%	48.5%	98.4%	1.6%	51.4%
	Vanilla AE	96.3%	91.7%	93.3%	0.6%	8.2%
	Latent AE-ND	95.4%	84.3%	99.8%	0.1%	15.6%
	Latent AE-NT	95.7%	93.2%	98.2%	1.7%	6.7%
	Latent AE	**98.2**%	92.5%	99.4%	0.5%	7.8%
Suppress	OCSVM	49.7%	6.6%	95.1%	4.8%	93.3%
	INDRA	74.3%	38.7%	96.3%	3.7%	61.2%
	Vanilla AE	84.3%	59.9%	99.9%	0.1%	40.0%
	Latent AE-ND	76.0%	41.3%	99.8%	0.1%	58.6%
	Latent AE-NT	85.4%	66.6%	97.5%	2.4%	33.3%
	Latent AE	**87.6**%	64.2%	99.9%	0.1%	35.7%
Flooding	OCSVM	48.8%	4.1%	96.2%	3.7%	95.8%
	INDRA	74.6%	44.2%	96.6%	3.4%	66.7%
	Vanilla AE	90.0%	74.5%	99.9%	0.1%	25.4%
	Latent AE-ND	87.6%	62.8%	99.8%	0.1%	37.1%
	Latent AE-NT	89.0%	77.3%	97.2%	2.7%	22.6%
	Latent AE	**91.3**%	75.9%	99.9%	0.1%	24.0%

SynCAN Dataset Attack Detection

In contrast to the ROAD dataset, where attacks targeted specific ID payloads for a set duration, the SynCAN dataset features attacks across various ID payloads with durations lasting 2-3 seconds, often with subtle changes closely resembling true signal values. Notably, flooding and suppress attacks alter ID transmission rates, constituting simple injection attacks. Table 7.5 presents the performance on the SynCAN dataset. OCSVM struggles to detect many attacks, likely due to slight value changes remaining within decision boundaries. Similarly, INDRA's reliance on signal-level intrusion scores hampers its ability to leverage feature associations, resulting in minimal reconstruction error for targeted signals. Vanilla AE surpasses both OCSVM and INDRA by exploiting feature associations, leading to higher reconstruction errors for attacks. However, Latent AE outperforms all models, including its variants, across all attacks. The effectiveness of Latent AE is present in Figure 7.10, where during the plateau attack, vanilla AE fails



(a) Reconstruction errors for vanilla and latent space AEs



(b) Reconstruction errors for Latent AE

Figure 7.10: Plateau attack reconstruction errors

to exceed the anomaly threshold, while latent space AE detects around 70% of the attack duration. Consequently, Latent AE identifies the majority of the attack window (Figure 7.10b). However, Latent AE variants do not exhibit promising detection performance for the SynCAN dataset similar to the ROAD dataset. This disparity arises because vanilla AE occasionally generates higher reconstruction errors, while latent space AE performs better in other instances, as depicted in Figure 7.11. This hinders Latent AE variants from surpassing vanilla AE. Although Latent AE outperforms all models, it may still miss some anomalies due to manipulated signals closely resembling actual values, which may not trigger anomalies in real vehicles as they do not alter benign CAN payload data. Additionally, SynCAN labels all frames within the attack period as anomalous, even though only a few are truly anomalous.

Figure 7.12 illustrates how variable association within the SynCAN dataset contributes to higher reconstruction errors. Specifically, Figure 7.12a shows the true and predicted values of feature id3_D3 during an attack, where the continuous drift from its true value

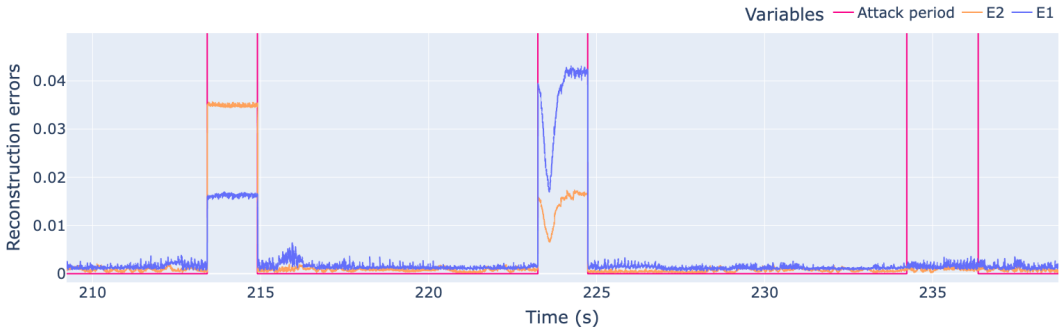
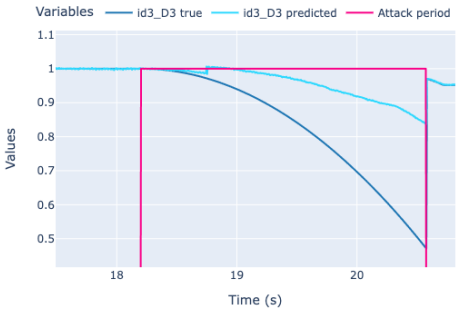
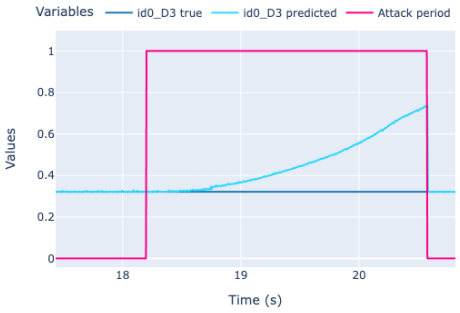


Figure 7.11: SynCAN reconstruction errors



(a) id3_D3 true and predicted values



(b) id0_D3 true and predicted values

Figure 7.12: SynCAN feature association

results in the AE's inability to accurately recreate the signal, creating a higher reconstruction error toward the end of attack period. Additionally, feature id0_D0 exhibits a high association with id3_D3, prompting the AE to adjust id0_D3 away from its true value to maintain the learned association, as evidenced in Figure 7.12b. These combined errors contribute to a higher reconstruction error at the message level. Moreover, this attack shows the anomalous value similarity to the true signal value at the beginning of the attack period which causes the higher FN rate for point level and small window-level attack detection. These experimental findings underscore the effectiveness of the proposed feature selection, transformed data structure, and enhanced AE, Latent AE, in detecting attacks on the CAN bus.

Table 7.6: Comparison of GRU, Latent AE and Ensemble IDS detection performance of ROAD dataset

Attack	Model	F1	TP	TN	FP	FN
Correlated signal	GRU	82.1%	83.8%	100%	0.0%	16.2%
	Latent AE	**100%**	100%	100%	0.0%	0.0%
	Ensemble IDS	**100%**	100%	100%	0.0%	0.0%
Correlated signal masquerades	GRU	86.7%	85.6%	100%	0.0%	14.4%
	Latent AE	**100%**	100%	100%	0.0%	0.0%
	Ensemble IDS	**100%**	100%	100%	0.0%	0.0%
Max speedometer	GRU	**100%**	100%	100%	0.0%	0.0%
	Latent AE	**100%**	100%	100%	0.0%	0.0%
	Ensemble IDS	**100%**	100%	100%	0.0%	0.0%
Max speedometer masquerades	GRU	**100%**	100%	100%	0.0%	0.0%
	Latent AE	**100%**	100%	100%	0.0%	0.0%
	Ensemble IDS	**100%**	100%	100%	0.0%	0.0%
Reverse light on	GRU	99.1%	99.0%	100%	0.0%	1.0%
	Latent AE	**100%**	100%	100%	0.0%	0.0%
	Ensemble IDS	**100%**	100%	100%	0.0%	0.0%
Reverse light on masquerade	GRU	99.4%	99.3%	100%	0.0%	0.7%
	Latent AE	**100%**	100%	100%	0.0%	0.0%
	Ensemble IDS	**100%**	100%	100%	0.0%	0.0%
Reverse light off	GRU	**100%**	100%	100%	0.0%	0.0%
	Latent AE	**100%**	100%	100%	0.0%	0.0%
	Ensemble IDS	**100%**	100%	100%	0.0%	0.0%
Reverse light off masquerade	GRU	**100%**	100%	100%	0.0%	0.0%
	Latent AE	**100%**	100%	100%	0.0%	0.0%
	Ensemble IDS	**100%**	100%	100%	0.0%	0.0%

Ensemble IDS Results

The ensemble IDS combines both the GRU model and the Latent AE model to enhance overall attack detection. For the ROAD dataset, both models utilise 25-millisecond windows, while for the SynCAN dataset, they employ 100-millisecond windows. Table 7.6 presents the detection performance of the ensemble IDS in comparison to the GRU and Latent AE models for the ROAD dataset attacks. The GRU model fails to detect the correlated signal and correlated signal masquerade attacks effectively with a high detection rate. Conversely, the Latent AE detects all attacks with a 100% detection rate. Since neither individual model generates false positives, the ensemble model matches the performance of the best individual model, which is the Latent AE, for the ROAD dataset.

Table 7.7 outlines the detection performance of the ensemble IDS in comparison to the GRU and Latent AE models for the SynCAN dataset attacks. In contrast to the ROAD attacks, the masquerade attacks in the SynCAN dataset do not change the CAN ID sequences. Consequently, the GRU model fails to effectively detect these attacks, as anticipated. However, flooding and suppress attacks alter the CAN ID sequences through frame injections and suspensions. Consequently, the GRU model successfully detects these two attacks with a 100% detection rate (TP). On the other hand, Latent AE

Table 7.7: Comparison of GRU, Latent AE and Ensemble IDS detection performance of SynCAN dataset

Attack	Model	F1	TP	TN	FP	FN
Plateau	GRU	46.0%	0.0%	100%	0.0%	100%
	Latent AE	**92.6**%	76.4%	99.3%	0.7%	23.5%
	Ensemble IDS	**92.6**%	76.4%	99.3%	0.7%	23.5%
Continuous	GRU	47.1%	0.0%	100.0%	0.0%	100%
	Latent AE	**95.4**%	84.1%	99.1%	0.8%	15.8%
	Ensemble IDS	**95.4**%	84.1%	99.1%	0.8%	15.8%
Playback	GRU	47.4%	0.0%	100%	0.0%	100%
	Latent AE	**98.2**%	92.5%	99.4%	0.5%	7.8%
	Ensemble IDS	**98.2**%	92.5%	99.4%	0.5%	7.8%
Suppress	GRU	**99.9**%	99.9%	99.9%	0.1%	0.0%
	Latent AE	87.6%	64.2%	99.9%	0.1%	35.7%
	Ensemble IDS	**99.9**%	100%	99.8%	0.1%	0.0%
Flooding	GRU	**100**%	100%	100%	0.0%	0.0%
	Latent AE	91.3%	75.9%	99.9%	0.1%	24.0%
	Ensemble IDS	99.9%	100%	99.9%	0.1%	0.0%

struggles to achieve a high detection rate for these two attacks, possibly due to minimal changes in signal values that do not generate significant reconstruction errors. However, the ensemble IDS achieves the performance level of the best individual model for all attacks. It is important to set anomaly and window thresholds to optimal values using benign datasets to minimize false positives. This is important to outperform the individual models.

The results obtained from the ROAD and SynCAN attack datasets shows the effectiveness of the ensemble IDS in detecting a diverse array of attacks with a higher detection rate. Although the Latent AE outperforms the GRU model for the ROAD attacks, the GRU model surpasses the Latent AE for two attacks in the SynCAN dataset. This highlights the importance of using an ensemble model rather than relying on individual models. By combining multiple models, the ensemble IDS enhances overall attack detection while mitigating the weaknesses inherent in individual models.

Comparison with Baseline Models

We conducted a comparative analysis between the proposed ensemble IDS and two baseline models: INDRA [101], a RNN-based model, and CANShield [66], a deep AE-based model developed with prior knowledge of the CAN specification. However, due to the unavailability of detailed information to replicate CANShield, we relied on comparing our results using the area under the curve (AUC) score reported in their paper. In the context of the ROAD dataset, masquerade attacks were not addressed in their findings,

Table 7.8: Comparison with Baseline Models - AUC Score

Dataset	Attack	Ensemble IDS	CANShield	INDRA
ROAD	Correlated signal	**1.00**	**1.00**	0.98
	Max speedometer	**1.00**	**1.00**	0.99
	Reverse light on	**1.00**	**1.00**	0.67
	Reverse light off	**1.00**	**0.99**	0.74
SynCAN	Plateau	0.93	**0.96**	0.72
	Continuous	**0.93**	0.87	0.79
	Playback	**0.97**	0.94	0.84
	Suppress	**0.99**	0.98	0.75
	Flooding	**0.99**	**0.99**	0.77

hence no comparison was made for those attacks. Our findings revealed that for the ROAD dataset, both the Ensemble IDS and CANShield successfully detected all attacks, with the exception of the reverse light-off attack, where CANShield achieved a slightly lower AUC score of 0.99. Regarding the SynCAN dataset, CANShield exhibited higher detection rates for the plateau attack. In the plateau attack, a single signal is overwritten to a constant value over a period of time. Since CANShield uses multiple AEs within the same time window, at least one AE is likely to detect these attack windows. The ensemble IDS outperformed CANShield for other attacks, except for the flooding attack, where both models showed identical detection levels. However, INDRA performed inadequately for both datasets due to its inability to detect contextual anomalies effectively.

Model Implementation on Raspberry Pi

To assess the computational overhead in a resource-constrained environment, we deployed the GRU and Latent AE models on a Raspberry Pi 4. Specifically, we utilised the Raspberry Pi 4 Model B 8GB version, along with a 16GB micro SD card. To optimize for on-device machine learning, we converted the trained GRU and Latent AE models into TensorFlow Lite (TFLite) versions. During this conversion process, we implemented quantization to minimize detection latency. We chose 16-bit quantization over 8-bit quantization as the latter slightly compromised accuracy in both models. Following conversion, the resulting TFLite models were named GRU-TFLite and Latent AE-TFLite. This conversion notably reduced the model size, with the Latent AE model shrinking from 1145 KB to 745 KB, and the GRU model decreasing from 233 KB to 49 KB. Subsequently, these TFLite models were deployed on the Raspberry Pi for further analysis of overhead.

The Raspberry Pi has the capability to simultaneously execute both the Latent AE-TFLite and GRU-TFLite lightweight models. However, running them concurrently results in a minor increase in processing time for each model compared to running them individually. During the inference process, streaming CAN data is stored in a buffer, as streaming CAN data is faster than data preprocessing. The ID-based model necessitates less time for both data preprocessing and inference when compared to the payload-based model. Given that both models employ the same window size T, the Ensemble IDS determines the window status by aggregating predictions from both the ID-based and payload-based models. Integrating the Raspberry Pi device into the CAN bus can be facilitated through interfaces such as OBD2-II or central gateways, effectively serving as an additional ECU.

Overhead Analysis

In addition to detection rate, detection latency, and memory consumption are critical aspects of a CAN IDS. Table 7.9 presents a comparison between payload-based and ID-based IDS in terms of the number of trainable model parameters, model size (in KB), and average inference time (in milliseconds). This analysis is conducted using the ROAD dataset and evaluated on a MacBook M1 Pro with 16 GB RAM. The reported number of parameters and model sizes are for a single model.

Among the considered models, INDRA stands out for its memory requirement and inference time, as it necessitates a separate model for each CAN ID. In contrast, among the payload-based models, Latent AE-ND offers optimal performance in terms of memory and inference time. This is achieved by eliminating the decoder component from the vanilla AE model. Latent AE requires only an additional 0.05ms for prediction compared to vanilla AE, resulting in an average inference time of 0.18ms per frame. The Latent space AE model is much smaller than the vanilla AE due to its limited number of input features and shallow model architecture. On the other hand, the GRU model exhibits significantly lower memory usage and higher efficiency than the payload-based models, relying solely on the CAN ID as input.

In Table 7.10, the average inference overhead, encompassing CPU utilisation, memory usage (RAM), and inference time, for the deployed TFLite models on the Raspberry Pi is presented. The ensemble IDS utilises 82% of the CPU and 94MB of RAM, with an inference time of 0.5ms per CAN frame. Despite this slight increase in inference time compared to the Latent AE-TFLite model, the ensemble IDS remains practical and

Table 7.9: Average detection latency and memory requirement

Features	Model	Parameters	Model size (KB)	Inference time (ms)
Payload	INDRA	190728	3200	0.44
	Vanilla AE	173731	882	0.13
	Latent AE-ND	87306	697	0.11
	Latent AE-NT	174489	1145	0.18
	Latent AE	174489	1145	0.18
ID	GRU	16945	233	0.08

Table 7.10: Average inference overhead on Raspberry Pi

Model	CPU (%)	Memory (MB)	Inference time (ms)
Latent AE-TFLite	38	58	0.4
GRU-TFLite	27	49	0.2
Ensemble IDS	82	94	0.5

deployable as an in-vehicle IDS due to its comprehensive attack detection capabilities. The outputs of the TFLite models on the Raspberry Pi closely align with the values obtained from TensorFlow models, with negligible accuracy differences. Therefore, the deployed models on the Raspberry Pi exhibit the same detection capabilities without any loss in accuracy. Overall, within a 25ms window, which encompasses around 50 CAN frames, the ensemble IDS requires only 25ms to provide window prediction. This enables the driver or the vehicle itself to take appropriate countermeasures in near real-time. Consequently, the proposed ensemble IDS proves suitable for detecting a wide range of attacks on the CAN bus in near real-time.

7.4.5 Limitations

For the payload-based model, normalization of each variable is imperative, typically based on the observed minimum and maximum values. However, if the training dataset lacks the true minimum and maximum values for each variable, any value surpassing the maximum or falling below the minimum during the inference stage could potentially trigger false positives. While employing a large and diverse training dataset can mitigate this limitation to some extent, certain variable values such as engine temperature might not reach their maximum values under normal driving conditions.

One potential drawback of black-box approach-based models, compared to signal value-based models, lies in the potential misalignment between payload features and signal values. In the context of a vehicle's CAN data specification, signal values can span multiple bytes with varying byte ordering or may be encoded within a single bit. Consequently, a payload feature could encompass multiple signal values, or conversely, multiple payload features may represent a single signal value. Despite this complexity, the Latent

AE approach is designed to discern patterns of feature associations within benign data and identify deviations from these patterns as anomalies. Even with byte-level feature fragmentation, Latent AE can still capture a significant portion of these association patterns. Disruptions caused by attacks on the CAN payload lead to notable increases in reconstruction errors, facilitating detection. Therefore, in our proposed method for attack detection, the impact of imprecise selection of actual signal boundaries is minimized if a payload feature maintains at least one highly associated feature. In other words, when a payload feature includes multiple signal values, as long as it has a highly associated feature, our method can detect association mismatches during an attack. Similarly, if a signal is represented by multiple payload features and any of these features exhibits a strong association, our method can identify the association discrepancy. In the ROAD dataset, approximately 98% of the features possess at least one highly associated feature. However, if a payload feature lacks a highly associated counterpart, it may lead to false negatives for the corresponding signals.

7.5 Conclusion

IDSs that exclusively utilise the CAN ID field effectively detect injection attacks and masquerade attacks that introduce new CAN ID sequences. However, time-series CAN payload data is crucial for identifying advanced masquerade attacks, which do not alter the ID sequences. Consequently, identifying a diverse range of attacks on the CAN bus is challenging and necessitates an IDS that employs multiple methods to cover a broad spectrum of attacks with limited computing resources.

Hence, we propose an ensemble IDS that integrates a GRU-based model with a novel AE model. Developing a CAN payload-based IDS poses challenges due to the lack of knowledge about CAN data specifications. Consequently, this work concentrates on leveraging raw payload values without decoding them into actual signal values, ensuring the proposed solution's adaptability across various car makes and models. Accordingly, The improved AE model, Latent AE, employs a novel feature selection method based on Cramér's $\hat{V}$ statistics and a transformed CAN payload data structure to handle the complexities of CAN data. Addressing the issue of high false negatives in vanilla AEs due to overgeneralization, Latent AE introduces a small latent space AE. Given that CAN bus payloads contain a higher number of categorical features with a limited number of unique values, vanilla AEs are susceptible to overgeneralization, potentially missing attacks on those variables. Experimental results demonstrate the efficacy of Latent AE

in near real-time detection of sophisticated attacks on CAN payloads, overcoming the limitations of vanilla AEs. The experiment results further indicate that the ensemble IDS enhances attack detection while mitigating the weaknesses of individual models. The proposed model incurs minimal inference overhead, making it suitable for deployment in real vehicles to detect various attacks in near real-time.

Chapter 8

A Comprehensive CAN Bus Attack Dataset from Moving Vehicles for Intrusion Detection System Evaluation

One of the major challenges identified in CAN Bus IDS research, as highlighted in Section 3.6, revolves around the limited availability of benchmark datasets with realistic and verified attacks. Consequently, the evaluation of model performance and the analysis presented in previous chapters heavily relied on the ROAD CAN intrusion dataset, which stands as the sole comprehensive attack dataset currently accessible. Despite acknowledging the significant limitations of the HCRL CH, HCRL SA, and SynCAN datasets, as discussed in Section 3.5, we utilised them to assess the generalization capabilities of the proposed models. In an effort to address this research gap in the field of CAN IDS research, we introduce a novel CAN bus attack dataset collected from a moving vehicle. This chapter helps to addresses the RQ2 and RQ3.

The key findings of this chapter have been accepted for publication in the Symposium on Vehicle Security and Privacy (VehicleSec) in the Network and Distributed System Security (NDSS) 2024.

Dataset	Real/ Synthetic	Attacks	Inj	Sus	Mas	Benign duration	Attack duration	Labeled
HCRL CH	Real	4	✓	-	-	0h 8m 20s	7h 21m 57s	Yes
HCRL OTIDS	Real	3	✓	-	✓	0h 17m 17s	0h 18m 56s	No
HCRL SA	Real	9	✓	-	-	0h 3m 31s	0h 8m 53s	Yes
HCRL CHDC	Real	4	✓	-	-	-	0h 23m 23s	Yes
SynCAN	Synthetic	5	✓	-	-	-	-	Yes
TU Eindhoven	Synthetic	5	✓	✓	-	0h 19m 20s	0h 8m 17s	Yes
ROAD	Real	13	✓	-	✓	3h 0m 32s	0h 27m 10s	No
CAN-MIRGU	Real	36	✓	✓	✓	17h 8m 10s	2h 54m 56s	Yes

Table 8.1: Publicly available CAN attack datasets. **Attacks:** indicating the count of distinct attack captures available in the dataset. For the SynCAN dataset, the duration of both benign and attack periods cannot be accurately determined using the provided timestamps. **Inj, Sus, Mas:** represent Injection, Suspension, and Masquerade attacks, respectively.

8.1 Introduction

Despite the recent increase in focus and publication of IDSs on the CAN bus [181, 34] the advancement of IDS research faces significant obstacles due to the lack of high-quality, publicly available real CAN data that includes realistic attack scenarios [30]. This is mainly due to the considerable cost and associated risks involved in generating real attack data on moving vehicles. The use of a real CAN dataset for model training, validation, and testing is crucial for the development of an effective IDS capable of detecting a wide range of attacks in real-world conditions. However, many proposed IDSs rely on self-collected datasets that are not accessible to other researchers [181]. Furthermore, the widely used HCRL CH dataset [72], despite being a popular public benchmark, has a significant drawback: benign data was collected during vehicle movement, while attack data was collected when the vehicle was stationary [34]. In an attempt to address these challenges, a more advanced dataset has been introduced in [30]. However, it is important to note that this dataset focuses on a limited number of IDs, and the vehicle was on a dynamometer during the collection of attack data.

To the best of the authors' knowledge, there is currently no publicly available CAN bus dataset that includes physically verified attacks collected during real-world driving conditions. In light of this gap, we present CAN-MIRGU, a real CAN bus dataset obtained from a modern automobile aiming to propel advancements in IDS research within in-vehicle networks. The comparative table for publicly available CAN attack datasets is presented in Table 8.1. This dataset is used to evaluate the CAN-ODTL and Latent AE IDSs.

8.2 Chapter Contribution

The primary contributions of this chapter can be outlined as follows:

1. Generating a CAN bus attack dataset while the vehicle is in motion under real-world conditions: This chapter introduces CAN-MIRGU, a novel and publicly available CAN bus benign and attack dataset collected from a modern automobile equipped with autonomous driving capability, operating under real-world driving conditions. This dataset includes physically verified attacks, addressing the existing gap in publicly accessible datasets featuring realistic attacks in dynamic driving scenarios.

2. Comprehensive training and testing dataset: The dataset includes 17 hours of benign data collected under diverse driving conditions to train IDSs with ample and varied data, enhancing their capacity to recognize normal driving behaviour. Moreover, it incorporates attack data with extended duration to assess IDS resilience under adversarial learning.

3. In-depth dataset analysis: This includes a thorough analysis of the dataset, offering insights to better understand both the benign and attack data. The availability of this detailed analysis provides valuable information for researchers and practitioners to gain a comprehensive understanding of the dataset's characteristics.

8.3 CAN-MIRGU dataset

This section details the experimental setup employed for collecting both benign and attack data in our dataset, named CAN-MIRGU. It outlines the procedures for the attacks, describes the vehicle's responses to each attack, and offers an analysis of both benign and attack data. The dataset is accessible through the following link: https://github.com/sampathrajapaksha/CAN-MIRGU.

8.3.1 Dataset collection setup

We utilised a modern automobile manufactured in 2016, and while we do not disclose the specific make and model, it is a fully electric vehicle equipped with full autonomous driving capabilities. To mitigate risks associated with executed attacks, the autonomous driving mode was deactivated, and professionally trained drivers were engaged for both

attack and benign data collection. The CAN data was captured using SocketCAN utilities[1] on a Linux laptop, employing the candump command. For data logging, a Kvaser Memorator 2xHS v2 was connected to the laptop using a standard USB 2.0 cable. In contrast to previous CAN data collection methods that involved connecting the CAN data logger directly to the OBD-II port [182, 93], we encountered limitations as only diagnostic messages were accessible through the OBD-II port of the vehicle in use. Consequently, the CAN data logger was directly connected to the CAN gateway to facilitate comprehensive data collection and injection. The candump speed was configured to 500 Kbps to align with the high-speed CAN bus. For injecting attack frames, Python-can[2] along with the cansend command in can-utils were employed, utilising Python 3.9.

For the benign data collection, the vehicle was driven mimicking the normal driving behaviour of an average driver on public roads in the UK to include various benign driving activities. Notably, these datasets were collected over a six-week period from 05/05/2023 to 12/06/2023, between 8am and 7pm, to account for normal variations in data due to diverse conditions and natural wear and tear of vehicle components. As a result, our benign dataset offers a more realistic representation of normal driving behaviours compared to other publicly available datasets.

Given the inherent risks of these injection attacks, the vehicle was driven at a maximum speed of 30 mph on the 750-acre proving ground belonging to Horiba MIRA during the attack data collection. Safety protocols were rigorously adhered to during the attacks, especially in situations affecting critical functions like steering. This attack dataset was collected from 24/10/2023 to 26/10/2023, between 11 am and 2 pm, considering the availability of the proving ground.

8.3.2 Attack scenarios

Injection attacks, including DoS, fuzzing, spoofing, and replay, were executed for specific IDs. One significant challenge with injection attacks is message confliction [9, 30]. As the attacker injects malicious frames, legitimate ECUs continue to send messages, leading to conflicts. The ECU's response to message confliction varies; simpler ECUs, like speedometers, might react based on straightforward algorithms, such as considering the last received message or utilising queuing algorithms. Complex ECUs, on the other hand, might choose to ignore conflicting messages or disable certain features if they are not safety-critical [9]. An effective approach to overwrite legitimate messages with the

[1] https://github.com/linux-can/can-utils
[2] https://python-can.readthedocs.io/en/stable/

target ID involves injecting malicious frames with the same ID immediately after the appearance of the legitimate frame, a technique known as flam delivery [30]. Consequently, we employed the flam delivery technique in the majority of our spoofing attacks.

To ensure a comprehensive evaluation of IDS across various IDs, we targeted five high-frequency IDs, including 2B0, 160, 251, 371, 372, five low-frequency IDs, including 07F, 50C, 559, 541, 593, and three medium-frequency IDs, including 381, 386, 394. These IDs are selected based on the prior knowledge of the CAN data specification. Prior to and after each injection attack, benign datasets were collected, allowing for the evaluation of IDS performance on both benign and attack data. Below are descriptions of the attack scenarios. Comprehensive details for each attack and message timing analysis are listed in Table 8.2 and Table A.1 in subsection 8.3.3. In these tables, the column labelled **Attacks** provides information on the attack name, injected ID and payload, attack duration in seconds, and the attack technique employed. The column labelled **Message Timing** displays the inter-message arrival time between all messages. Here, the x-axis represents time in seconds, and the y-axis represents inter-message arrival time in milliseconds (ms). The column **Targeted ID Message Timing** illustrates the transmission of frames for the injected ID near the attack start, with the attack area shaded. The x-axis represents the inter-arrival time for the injected ID, while the y-axis represents CAN IDs, using two same-frequency CAN IDs for comparison. For DoS and fuzzing random IDs attacks, where no specific ID was targeted, the same ID (340) is utilised for comparison purposes.

DoS attack

Given that CAN ID 0x000 was not a valid ID for this vehicle and considering it is the highest priority ID, it was utilised to perform the DoS attack with the maximum payload (FFFFFFFFFFFFFFFF). Frames with ID 0x000 were injected every 0.001s. However, no attack reactions were observed during the attack period. This lack of response could stem from CAN ID 0x000 not being a valid ID for this vehicle or possibly from a violation of the checksum mechanism used by this vehicle.

Fuzzing attack

There are two variations of the fuzzing attack, each performed with different IDs. In the fuzzing attack with random IDs, an ID was randomly selected from the range of 0x000 to 0x255 and injected with the maximum payload, similar to the DoS attack, which is a valid payload according to the CAN specification. This led to the observation of a few

warning lights on the dashboard and occasional warning sounds. In the other variation, randomly selected valid IDs were injected with the maximum payload, resulting in more warning messages. For both variations, frames were injected every 0.02s.

Replay attack

This dataset includes three replay attacks, where previously transmitted payloads were injected into unusual contexts using flam delivery. In these instances, the injected frames were placed in situations or sequences that deviated from their original context or intended use. These include steering angle replay attack, Engine Management System (EMS) replay attack, and EMS replay long attack. No visible changes were observed during the replay attacks.

Spoofing attacks

The majority of the attacks in the CAN-MIRGU dataset are spoofing attacks. Both flam delivery and time-based injection were employed for different attacks depending on the targeted ID and payload. These attacks encompass various scenarios such as steering angle, brake and fog light, brake warning, drive mode changing, Forward Collision Avoidance Assist (FCA) warning, power steering, max speedometer, three variations of min speedometer, wiper warning, EMS, parking brake, two variations of gear shifter, and door open warning attacks. All of these attacks involve a single attack window that spans over a few seconds or a few minutes. Additionally, there are four attack datasets with multiple attack windows, including fuzzing valid IDs and DoS attacks with two attack windows, reverse speedometer and fuzzing attacks with two attack windows, and two variations of multiple attacks with three and six attack windows.

Suspension attack

Using small benign datasets, we simulated five suspension attacks by removing legitimate target ID frames for a specific period of time. This simulation replicates the suspension of an ECU. The selected IDs for these attacks are 160, 371, 386, 541, and 07F, covering high, medium, and low-frequency IDs. While real suspension attacks may alter other payload values due to variable associations, this change is not reflected in the simulated attacks. Nevertheless, despite this limitation, these attacks remain effective for testing the detection latency of the initial attack instance, critical for prompt detection.

Masquerade attacks

This was simulated by employing five selected real spoofing attack captures that utilised flam delivery as the attack technique. Similar to the approach used in [30], we removed the legitimate target ID frames preceding each injected frame to create more advanced versions of the attacks. This approach eliminates message confliction in the data, creating the appearance that only the spoofed messages are present during the injection interval. The selected spoofing attacks used to produce masquerade attack versions are break warning, steering angle, wiper warning, min speedometer, and break and fog light attacks. While masquerade attacks are simulated through post-processing, the impact of the malicious frames employed in these attacks was physically verified during real attacks. Like suspension attacks, real attacks may introduce additional changes not captured in simulated versions. However, these simulations remain effective for testing the detection latency of the initial attack instance.

Table 8.2, Table A.1, Table A.2 and Table A.3 present a comprehensive summary of attacks along with visualizations of message timing. These visualizations illustrate the inter-message arrival times between all messages and the transmission of frames for the injected messages. They provide insights into how the malicious frames impact these times based on the attack technique, whether flam or time-based injection. The targeted ID message timing plots reveal changes only in the transmission of injected frames, while other ID transmissions remain unchanged. Therefore, frequency or time-based IDSs should leverage ID-level information for enhanced detection rates. For masquerade attacks, the transmission of targeted ID messages mirrors that of benign messages, posing a challenge for time-based IDSs, which might struggle to detect these attacks (see the targeted ID message timing for break warning masquerade attack). Conversely, in the case of suspension attacks, the targeted ID's frame transmission halts during the attack period, as depicted in the targeted ID message timing for the ID 160 suspension attack. These findings offer valuable insights for designing IDSs capable of detecting attacks with lower latency and higher detection rates.

8.3.3 Benign and attack data analysis

CAN-MIRGU dataset comprises 26 real injection attacks and 10 simulated attacks for suspension and masquerade attacks, totalling 36 attacks that targeted 13 IDs out of the total 56 CAN IDs. The real injection attack captures span over a duration of 2 hours, 9 minutes, and 16 seconds, while suspension attacks span for 26 minutes and 16 seconds,

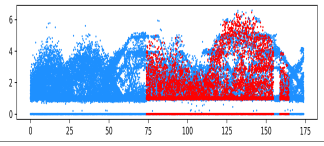
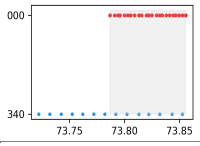
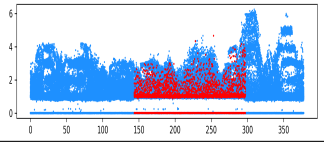
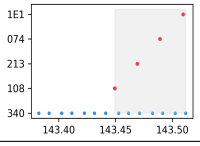
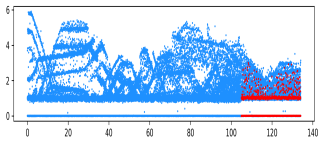
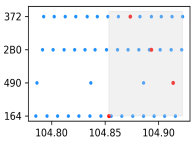
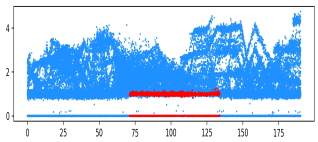
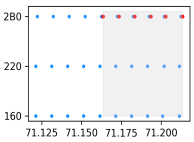
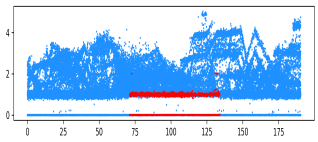
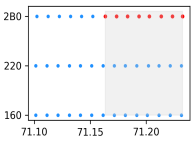
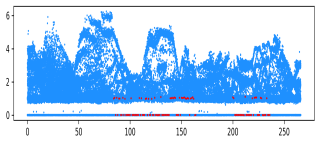
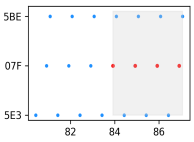
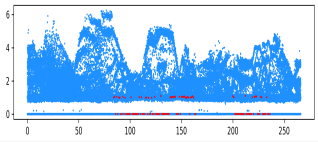
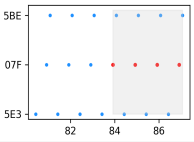
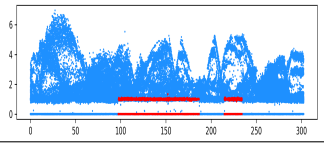
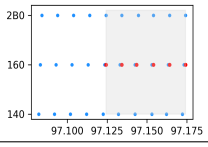
Attack	Observations	Message Timing	Targeted ID Message Timing
DoS ★ 000#FFFFFFFFFFFFFFFF 153.704584 Injecting every 0.02s	No visible changes.		
Fuzzing random IDs ★ XXX#FFFFFFFFFFFFFFFF 153.704584 Injecting every 0.02s	Few warning lights on the dashboard and occasional warning sounds.		
Fuzzing valid IDs ★ XXX#FFFFFFFFFFFFFFFF 134.016241 Injecting every 0.02s	'Check FCA (Forward Coll. Avoidance Assist)' warning message, parking brake, and ABS indicators on the dashboard. 'Harness Relay Malfunction' warning message on the lane detection display and continuous warning sounds.		
Steering angle ★ 2B0#XXAAXXXXXX 190.264094 Flam	'Check FCA(Forward Coll Avoidance Assist)' warning message on the dashboard and continuous warning sounds.		
Steering angle masquerade ★ 2B0#XXAAXXXXXX 190.264094 Masquerade	Synthetic attack. It is likely to have a similar observation to the steering angle real attack.		
Break and fog light ★ 07F#XXC3XXXXXXXXXXXX 266.008802 Flam	'Check brake light' and 'Check fog light' warning messages on the dashboard and continuous warning sounds.		
Break and fog light masquerade ★ 07F#XXC3XXXXXXXXXXXX 266.008802 Flam	Synthetic attack. It is likely to have a similar observation to the break and fog light real attack.		
Break warning ★ 160#02AAXXXXXXXXXXXX 266.008802 Flam	'Stop vehicle and check breaks' warning message on the dashboard and continuous warning sounds.		

Table 8.2: Description of attacks. In columns, **Message Timing** and **Targeted ID message Timing**, blue dots and red dots indicate benign and attack frames, respectively. Severity of the attack is categorized with ★ for no impact, ★ for warnings, and ★ for significant behaviour alteration.

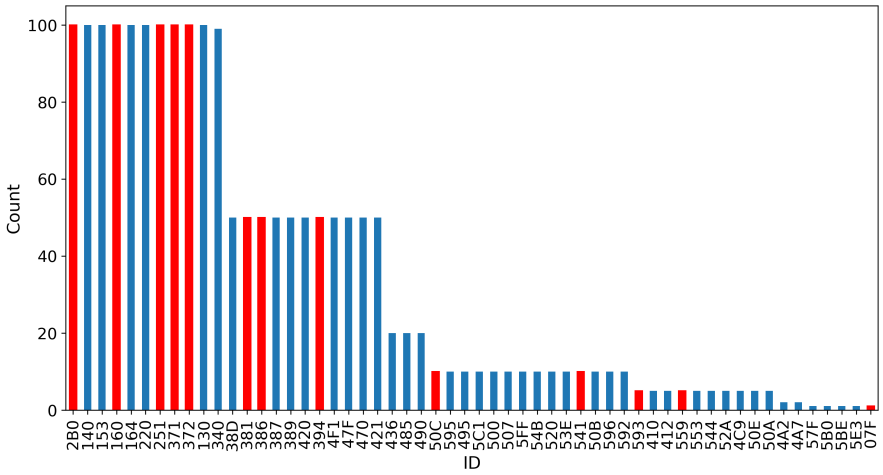


Figure 8.1: Average number of ID counts for one-second driving. Targeted IDs for attacks are shown in red bars.

and masquerade attacks span for 19 minutes and 24 seconds. Additionally, the dataset includes 17 hours of benign data, providing a substantial dataset for training an IDS to learn the normal behaviour of the vehicle. The average number of ID counts for one second of benign driving data is illustrated in Figure 8.1. The IDs selected for the attacks are highlighted in red bars. Based on this, the selected targeted IDs range from the highest frequent ID 2B0 to the lowest frequent ID 07F. This facilitates the evaluation of IDS performance against different frequent IDs, as detection capability might depend on the characteristics of each ID. Figure 8.2 depicts the frame transmission of normal driving over one second for the targeted 13 IDs. These frequent patterns are expected to change during the injection and suspension attacks due to the introduction of additional frames or suspension of frames. In contrast, masquerade attacks do not change this pattern, as they do not introduce any new frames.

All CAN data files are logged using the candump command in can-utils. In addition to the standard fields of candump, labels are assigned as 0 for benign frames and 1 for attack frames, as illustrated in Figure 8.3.

For the benign datasets, all labels are set to 0 since there are no instances of attack frames present. For suspension attacks, where frames associated with the targeted ID are removed throughout the attack period, leading to the absence of malicious frames, the entire attack window is labelled as 1. This labelling is necessary as IDSs must identify the attack window in the context of suspension attacks. Metadata is provided in JSON format

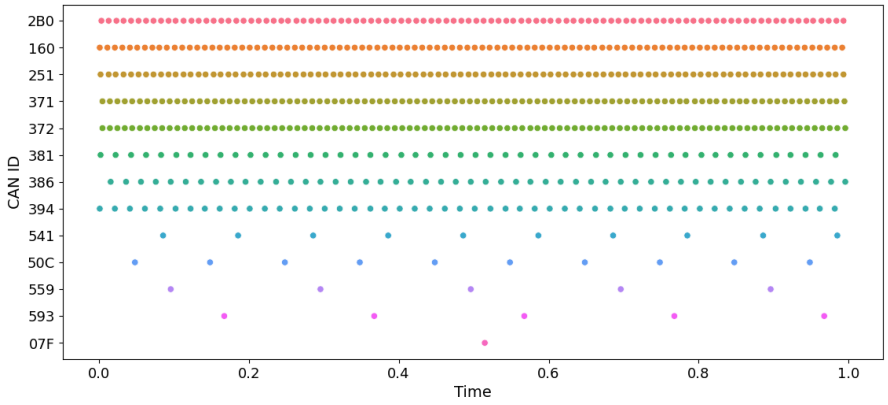


Figure 8.2: Frame transmission over one second for the targeted IDs

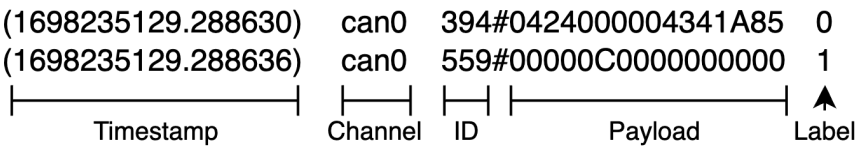


Figure 8.3: CAN bus data format

for each attack capture, including the attack name, description, the length of the capture in seconds, attack duration in seconds, injected ID and payload, injection interval (start, end), attack type (real or synthetic), and attack technique (flam or time-based injection). An example of the provided metadata is shown in Figure 8.4. In the injection_data_str of the metadata, the wildcard character 'X' is used to indicate that these positions are not changed during flam delivery. Instead, the payload values of the last transmitted same ID are used for these positions. Similarly, 'X' in the injection_CAN_id field in fuzzing attacks indicates that no particular single ID is targeted. Random numbers are used in fuzzing random ID attacks, and a set of valid IDs are used in fuzzing valid ID attacks.

8.4 Discussion

Our attacks targeted 13 IDs based on ECU functionalities. For instance, the steering angle attack targeted the ID associated with steering-related data, while the EMS attack targeted the ID associated with the engine management system. However, for certain attacks, we did not observe any noticeable changes. This lack of observation could be attributed to potential changes that are not visible or the CAN bus actively ignoring

```
"Break_warning_attack":{
 "description": "normal driving; start injecting; 'Stop vehicle and check breaks'
  warning message and sound; stop injecting; normal driving",
 "elapsed_sec":302.424109,
 "attack_duration":137.053789,
 "injection_data_str":"02AAXXXXXXXXXXX",
 "injection_CAN_id":"0x160",
 "injection_interval":[
    1698233312.889207,
    1698233449.942996
 ],
 "attack_type":"real",
 "attack_technique":"flam",
},
```


Figure 8.4: Snapshot of metadata for one attack

inconsistent messages, possibly as a safety measure [9]. Time-based injection attacks targeting high-frequency IDs, such as fuzzing valid IDs and max-min speedometer attacks, often resulted in bus-off situations. In the creation of the ROAD dataset, the use of the maximum payload during fuzzing attacks aimed to prevent accidental ECU bus-offs [30]. Nevertheless, our experiments indicated that the occurrence of bus-off situations primarily depends on the injection frequency rather than the payload used. Consequently, we mitigated this issue by reducing the injection frequency based on repeated experiments. In cases of bus-off, we had to disconnect and reconnect the CAN data logger to re-establish the connection. It is worth noting that a bus-off situation was observed for drive mode changing attacks, which targeted a low-frequency ID, 50C. In the DoS attack, the message timing plot in Table 8.2 shows a short period within the attack window where no attack frames are available, while frames with ID 0x000 continue to inject continuously throughout the attack period. This pattern is also evident in break warning, break and fog lights attacks, indicating a potential tendency of the CAN bus to temporarily ignore malicious frames.

The reactions to the majority of attacks were observed as warning messages, illuminated dashboard lights, and continuous warning sounds. While most attacks triggered non-critical responses, some can be classified as safety-critical, posing risks to the vehicle and its passengers. Max and min speedometer attacks, focusing on ID 386 associated with four-wheel speeds, share similarities with the correlated signal attack aimed at manipulating the speeds of the four wheels in the ROAD dataset [30]. In the ROAD dataset, this attack led to the immobilization of the car due to varying, unrelated speeds among

the wheels. By injecting different speed values into the respective bytes of the payload, max and min speedometer attacks did not result in physical changes to the vehicle but displayed inaccurate speedometer values. Nevertheless, this could be exploited by adversaries to deceive drivers, particularly in speed-regulated areas, posing potentially serious consequences. Two gear shifter attacks targeted the ECU associated with gear control, each employing distinct payloads. In the first case (payload: 800001000000AA05), a 'Shifting not possible due to overheating' warning message continuously appeared on the dashboard, accompanied by warning sounds. Simultaneously, the driver experienced a stiff steering wheel, necessitating significant force to turn. In the second case (payload: 000001000000AA05), with only a slight change in the first nibble of the payload, the same warning message and sounds were present, but the steering exhibited looseness, making the vehicle overly responsive to minimal steering adjustments. Both attacks posed a risk of losing control over the vehicle. The noteworthy aspect is that this vehicle is equipped with full autonomous driving capabilities. Given that the vehicle is trained to navigate based on the curvature of the road, aided by the lane detection system, attacks of this nature on an autonomous vehicle could result in the vehicle deviating from its lane. Targeting the drive mode-associated ID 50C led to continuous switching between normal, sport, eco, and eco+ driving modes, resulting in unstable vehicle behaviour and jerking. However, the attacker node entered a bus-off mode shortly after the attack, a safety measure that, while preventing prolonged damage, still allowed for a potentially significant impact during the attack duration.

Certain vehicle functions require input from multiple CAN IDs with specific data to activate the functionality [9]. This was evident in the wiper warning attack. During this attack, we specifically altered the nibble of the payload associated with wiper position 2. However, despite the display on the dashboard indicating that the front wiper was set to level 2, there was no physical movement of the wiper. This occurrence suggests that additional changes to other associated IDs with specific data values may be required to activate the actuators. Some of these results can be observed in the demonstration video available at https://youtu.be/CufiACr2Zs8

There is a lack of publicly available datasets for evaluating CAN IDSs against adversarial attacks. To address this gap, we have included a comprehensive set of attacks, including EMS replay long and two multiple attack captures (multiple attack 1 and multiple attack 2), designed for assessing IDS resilience against the model and data poisoning attacks. Despite the absence of visible changes during EMS replay attacks, these types of attacks can be leveraged by adversaries to poison training datasets. Consequently, it is crucial

to evaluate IDS resilience under adversarial learning conditions.

As the CAN-MIRGU dataset incorporates unaltered raw CAN data for both benign and attack instances, it is suitable for testing a range of IDS. This allows for the evaluation of IDSs employing various features, including timing, ID sequences, and payload data. It is important to highlight that the intended alterations to vehicle functionality were physically verified for all included injection attacks. For almost all injection attacks performed, the observations were instantaneous. Therefore, any IDS designed for the CAN bus should prioritize detecting the first instance of an attack within the shortest possible time. This focus on detection latency is crucial for implementing prompt countermeasures.

While this dataset offers notable advantages, there are certain limitations to consider. The maximum speed of the vehicle during attack collection was 30 mph. Although the benign dataset encompasses driving scenarios at various speeds including 30 mph, executing attacks for other higher speeds used in benign driving was not feasible. Additionally, our simulated masquerade and suspension attacks may differ from real attacks.

8.5 Conclusion

Despite the recent surge in focus and publication of IDSs for the CAN bus, advancing IDS research encounters significant hurdles due to the absence of high-quality, publicly available real CAN data that incorporates realistic attacks. This is mainly due to the substantial cost and associated risks linked to generating real attack data on moving vehicles.

To overcome this challenge, we presented a novel and publicly available CAN bus attack dataset collected from a modern automobile equipped with autonomous driving capabilities operating under real-world driving conditions. This dataset encompasses physically verified attacks, effectively filling the existing gap in publicly accessible CAN datasets featuring realistic attacks within dynamic driving scenarios. This, in turn, facilitates the thorough testing of various techniques presented in the literature. The availability of this dataset promises to enhance the comparison and validation of proposed IDS solutions. In the next chapter, we utilise this dataset to evaluate the IDSs proposed in this thesis.

Chapter 9

Model Deployment

The CAN ID-based IDS, CAN-ODTL, which incorporates the streaming learning capabilities introduced in Chapter 6, and the Latent AE, an improved AE-based IDS introduced in Chapter 7, demonstrated promising results on publicly available datasets, namely ROAD, SynCAN, and HCRL CH and HCRL SA. Additionally, experiments conducted on the resource-constrained Raspberry Pi device revealed their deployability to monitor CAN data and detect injection, suspension, and masquerade attacks with near real-time performance. However, it is crucial to deploy these IDSs into a real vehicle and conduct a comprehensive evaluation to assess their effectiveness in real-world settings. Accordingly, this chapter discusses the model deployment and evaluation, utilising the dataset introduced in Chapter 8. This chapter addresses the RQ4.

9.1 Introduction

IDSs which utilised various fields of CAN frame are discussed in Chapter 3. The majority of these proposed IDSs have a primary emphasis on enhancing the efficacy of attack detection. However, it is imperative to focus towards improving the detection latency as well, given the CAN bus's transmission of a substantial number of frames per second. Considering that potential attacks may target safety-critical systems within vehicles, an IDS must demonstrate the capability to swiftly detect these attacks. This rapid detection ensures the prompt initiation of appropriate countermeasures to minimize the potential impact of such attacks.

However, assessing the attack detection capability and detection latency of these IDSs

proves to be challenging without their deployment in an actual vehicle. This challenge stems from the technical expertise required to effectively implement these models in a real vehicle, and addressing the complexities associated with attacking a moving vehicle, as explained in Chapter 8. In order to address these challenges, we collaborate with our industry partner Horiba MIRA Ltd., an automotive engineering and development consultancy based in the UK with an expansive 850-acre technology park. We utilised their proving ground for attack data collection and IDS functionality testing. The process of benign and attack data collection is detailed in Chapter 8, while this chapter focuses on the deployment of the models for continuous monitoring of CAN data and the performance evaluation of the CAN-ODTL and Latent AE models using the collected attack dataset.

9.2 Chapter Contribution

The primary contribution of this chapter can be outlined as follows:

1. Improving CAN-ODTL and Latent AE for near real-time attack detection: This chapter further improves the CAN-ODTL and Latent AE models, specifically focusing on enhancing near real-time detection capabilities while maintaining a high attack detection capability.

2. CAN-ODTL retraining: The improved CAN-ODTL model is retrained with an extensive benign dataset on a Raspberry Pi device by playing real CAN data logs, simulating actual CAN transmission.

3. Model deployment and evaluation: The retrained CAN-ODTL and Latent AE models are integrated into the Raspberry Pi and connected to the vehicle through the CAN gateway for continuous CAN data monitoring and are evaluated with different attacks.

9.3 IDS Improvements

To achieve near real-time detection latency in the deployed environment, enhancements were necessary for both CAN-ODTL and Latent AE models. This section outlines the requirements for these improvements and details the implemented improvements.

9.3.1 CAN-ODTL Improvements

The vehicle selected for the deployment experiments includes 56 CAN IDs and transmits around 2000 frames per second, resulting in a frame transmitted every 0.5ms. As discussed in Chapter 8, the benign dataset includes 17 hours of driving data collected under diverse driving scenarios over a 6-week period. Since CAN-ODTL requires pre-training with a substantial amount of benign data, four hours of benign data collected in the first week are considered for the initial model training (pre-training). As discussed in Chapter 5 and Chapter 6, this model is optimized to predict the centre ID accurately given the pre and post-context. Accordingly, a window size of 11 is selected, with 5 IDs assigned to each as pre and post-context. Since the attacks might target multiple IDs representing high, medium, and low-frequency IDs, it is important to achieve higher accuracy for all IDs to enhance attack detection for various IDs. A one-hour dataset is used to evaluate the centre ID prediction accuracy using the trained model. The results indicate higher centre ID prediction accuracy for the majority of high-frequency IDs, whereas low-frequency IDs achieved relatively lower accuracy. This discrepancy may arise from two reasons.

Firstly, having high, medium, and low-frequency IDs creates an imbalanced dataset, where the model is biased towards high-frequency IDs, leading to higher accuracy for them and lower accuracy for medium and low-frequency IDs. The ID distribution for this vehicle is shown in Figure 8.1. For a fixed dataset, it provides more training samples for high-frequency IDs and fewer training samples for low-frequency IDs. This underscores the importance of implementing multiple models focusing on different subsets of IDs and performing retraining for each model. For instance, high-frequency IDs might not need retraining with a very large dataset, whereas medium and low-frequency IDs might require retraining with a substantially larger dataset collected over a few weeks or months. The second reason is that some IDs may be easier to predict as they might always follow a limited set of IDs, while others might follow a large number of IDs. Therefore, retraining is pivotal for IDs that achieve lower ID prediction accuracy.

To evaluate the detection latency of the IDS under realistic CAN data transmission, before deployment, we replayed the one-hour dataset using the canplayer available in can-utils on the Raspberry Pi 4 model B 8GB version. This facilitated the replay of the collected CAN logs with the maximum bitrate of the CAN bus. The CAN-ODTL IDS was converted into a TFLite version using float 16 quantization, integrated into the Raspberry Pi, and used to predict the IDs while canplayer played the CAN logs. This

virtual CAN network replicated real CAN data transmission before deploying the model into the vehicle.

The results show that, on average, the IDS can predict a frame in around 0.34ms, achieving near real-time detection considering the 0.5ms data transmission rate of the CAN bus. However, since our objective is to integrate both CAN-ODTL and Latent AE models on the Raspberry Pi, running both models in parallel on the resource-limited Raspberry Pi device increases this time, leading to an increase in detection time. It is pivotal to achieve a detection time below the 0.5ms data transmission rate. Even a very small delay in detection is not appropriate in this case. For example, we observed that running both CAN-MIRGU and Latent AE models achieved around 0.56ms on average. This is only a 0.06ms delay. However, since it transmits around 2000 frames per second, this delay accumulates with the number of frames and creates a significant delay after running for a few minutes. Therefore, it is necessary to improve both CAN-ODTL and Latent AE IDSs to enhance detection latency and accuracy. Our experiments with various context window sizes showed that the size of the context windows significantly affects the CAN-ODTL IDS detection latency.

9.3.2 Latent AE Improvements

Similar to the CAN-ODTL model evaluation, we train and evaluate the Latent AE model on the Raspberry Pi while playing the CAN dump with canplayer. The model is trained based on the methods discussed in Chapter 7. This showed that it can provide inference for one frame in 0.41ms, which is below 0.5ms, providing near-real-time detection. However, as discussed earlier, running CAN-ODTL and Latent AE in parallel resulted in an increased detection latency of 0.56ms. Therefore, similar to the CAN-ODTL IDS, it requires improvement for the Latent AE IDS. Our experiments show that training one model for each ID significantly improves the detection latency. Latent AE proposed in Chapter 7 trains one model for all IDs using the selected payload features. For unassociated variables of each ID, this used zero padding. As a result, this required a small latent size and helped to make the model lightweight. However, removing these unassociated features for each ID without using zero padding facilitates training separate models for each ID. This makes each model significantly lighter and results in improved detection latency. This utilises PCA to select the appropriate latent size. Therefore, it does not need to train the models for each ID manually. Instead, it can train the model for each ID by selecting associated features (feature columns) for each ID. During the inference process, to optimize data preprocessing, a Python dictionary is maintained to update

the latest payload for each CAN ID. Then, only the payload for the current ID and its associated variables are extracted for prediction using the model trained for that specific ID. This approach makes data preprocessing efficient and requires minimal data to be stored in memory during inference.

9.3.3 Experimental setup

Since CAN-ODTL detection latency depends on the selected context size, to reduce the waiting time for prediction, we set the post-context to 5 IDs and experimented with various pre-context sizes, including 5, 10, 15, 20, 25, and 30. This led to the selection of multiple models with different context windows to improve both detection latency and accuracy. Accordingly, three models were trained by grouping IDs into three groups based on their ID prediction accuracy. These three models use 5, 15, and 30 as the pre-context size while all use 5 as the post-context size. Low frequent IDs required a longer context to achieve higher accuracy. However, this did not result in higher average detection latency as these are not frequently present in CAN data. On the other hand, most high frequent IDs required only 5 as the pre-context, which led to a significant improvement in detection latency. Only the model that uses 30 as the pre-context showed improvement with the use of a self-attention layer, whereas the other two models did not show improvement with the self-attention layer. Models with 5 and 15 pre-context IDs use 25 nodes in the embedding layer and 16 nodes in the GRU layer. The model with 30 pre-context IDs required 40 nodes in the embedding layer, 32 nodes in the GRU layer, and a self-attention layer to achieve higher attack detection. Grid search is used to optimize these hyperparameters.

In the Latent AE feature selection process, we employ Cramér's V statistic to select the five most highly associated features for each payload variable of a specific ID. If an ID has fewer variables than 8 bytes, zero padding is applied to account for missing variables, resulting in each ID having 48 variables. This approach allows models to be trained with the same parameters without manually selecting variables for each ID model. Selecting a higher number of associated variables increases model complexity and improves attack detection, while selecting fewer variables reduces model complexity and decreases attack detection. To strike a balance between these factors, we set this value to 5 based on repeated experiments with different values. The Latent AE model utilises a symmetric AE architecture for both AEs. The encoder is constrained to have 2 hidden layers, including the latent layer, while the Latent space AE includes only the latent layer as the hidden layer. For each AE model trained for each ID, the first hidden layer is set to

64 nodes, and PCA is employed to determine the latent space size. Specifically, the latent space size for the encoder and Latent space AE is determined based on capturing 90% and 99% of the variability, respectively. For both CAN-ODTL and Latent AE models, the window size is set to 0.25ms to give the window prediction. All CAN-ODTL and Latent AE models are converted into TFLite versions with float 16 quantization to improve the inference on Raspberry Pi.

9.3.4 Model Retraining

Given the necessity for a substantial dataset to retrain the CAN-ODTL IDS, the retraining process involves utilising the collected benign dataset. As previously mentioned, a four-hour dataset is employed for the model pre-training. Throughout the retraining phase, a separate one-hour dataset is used to monitor the model's progress to identify any potential overfitting, while another two-hour dataset is used to estimate the thresholds. The remaining dataset is utilised for IDS retraining, accomplished by replaying the CAN logs using the canplayer on the Raspberry Pi. The decision to conduct retraining experiments on the Raspberry Pi while playing the CAN logs was due to the limited time allocated for the deployment experiments. However, similar experiments conducted during the deployment for a brief duration reveal that the retraining efficiency aligns with the experiments performed on the Raspberry Pi by playing the CAN logs. Consequently, these retraining procedures can be extended over a more extended time period in the real deployment. Careful selection of the learning rate decay and momentum parameters of the stochastic gradient descent (SGD) optimizer is crucial to mitigate overfitting. To address this concern, we set the initial learning rate to 0.0001 and decreased it by a factor of 0.99 every 100 batches, with momentum set to 0.99. These values were chosen based on repeated retraining experiments, aiming to prevent overfitting specifically during the retraining of the classification layer. During the retraining, the batch size is set to 256.

9.4 Deployment of Models on the Vehicle

Improved CAN-ODTL and Latent AE models are saved in the Raspberry Pi to make the inference for streaming CAN data. Usually, OBD-II gives access to the CAN bus of the vehicle. However, the vehicle which was used for the experiments only gives access to the diagnostic messages through the OBD-II port. Therefore, the Raspberry Pi is connected to the CAN bus directly through the CAN central gateway. This allows both message injection and CAN data (CAN dump) collection. The equipment setup for this

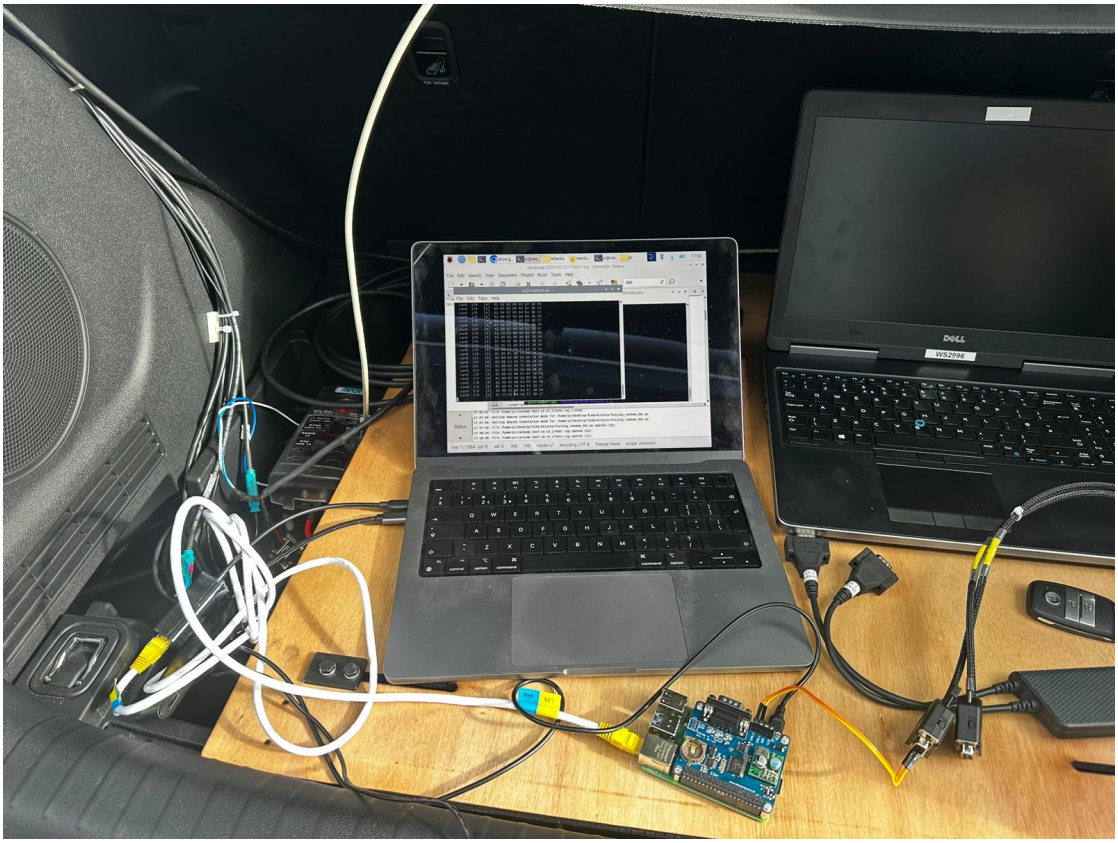


Figure 9.1: Model deployment equipment setup

deployment is shown in Figure 9.1. Raspberry Pi was connected to a Macbook M1 Pro with 16GB RAM to monitor and control the Python scripts. This deployment is similar to connecting an additional ECU to the CAN bus for CAN data monitoring and can be classified as a network-based IDS deployment. This is depicted in Figure 9.2.

9.5 Evaluation and Performance Results

This section outlines the CAN-ODTL model selection, retraining progress and evaluates the performance of both CAN-ODTL and Latent AE models against the executed attacks.

9.5.1 CAN-ODTL Model Selection

To accommodate the necessity of several models with different context sizes, we conducted experiments with various pre-context sizes to determine the optimal context sizes and groupings of CAN IDs for training with specific context sizes. The post-context

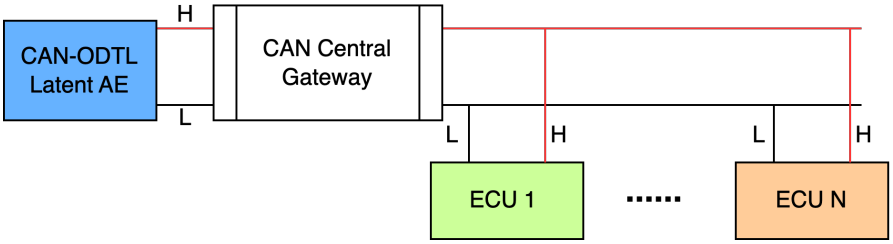


Figure 9.2: Model Deployment on the CAN bus

size was set to 5 IDs. After experimenting with different pre-context sizes, the average prediction accuracy was considered to identify the best context and group the IDs. The average ID prediction accuracies for each ID are depicted in Figure 9.3. Some IDs could achieve higher ID prediction accuracy even with a small pre-context size, while others exhibited lower ID prediction accuracy. Consequently, the IDs were grouped into three categories to train three models. The first group consists of 19 CAN IDs that required only a few pre-context IDs to achieve a high prediction accuracy of over 0.98, primarily comprising high-frequency IDs. The second group contains 20 CAN IDs with prediction accuracies ranging from 0.8 to 0.98. The third group includes 17 CAN IDs, mostly low-frequency IDs, that achieved a prediction accuracy below 0.8. Following the selection of these groups, each model was trained to achieve higher ID prediction accuracy by selecting the best hyperparameters, such as pre-context size and the number of nodes in each layer, through grid search.

9.5.2 CAN-ODTL Model Retraining

Each model undergoes retraining using the available benign dataset, which spans over six weeks and includes multiple CAN logs, with each log containing around 1-2 hours of data. The retraining process incorporates the data poisoning defence procedure to filter out potential poison windows before retraining. Monitoring the retraining progress is crucial to identify potential overfitting. Hence, after every CAN data log retraining, we assess the progress. Since the model is trained to predict IDs with higher accuracy given the pre and post-context, it should exhibit higher softmax probabilities when predicting each IDs. Therefore, for each ID, the softmax probability distribution should show a negative skew with retraining. In Figure 9.4, the change in softmax distribution for ID 50C during retraining is illustrated. However, visually tracking these changes for each ID individually during model retraining poses challenges. To address this, we use average positive and negative skewness to monitor the overall model progress. Larger negative

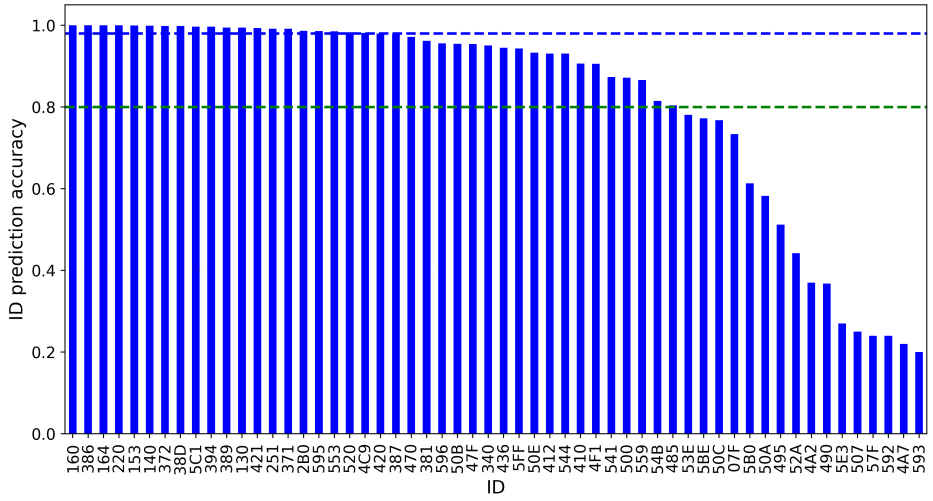
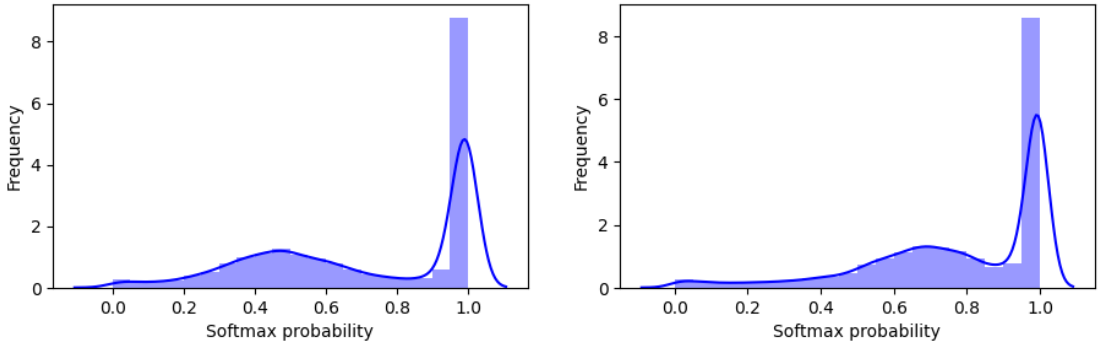


Figure 9.3: ID prediction accuracy. Blue and Green lines represent the thresholds used to classify IDs into three distinct groups

skewness is expected to indicate higher accuracy in predicting each ID. The positive and negative skewness for both the pre-trained and re-trained models, which include low-frequency IDs, are displayed in Figure 9.5. According to this, retraining proves beneficial in achieving better predictability for each ID. During retraining, after each CAN data log, we compare the average skewness for both positive and negative skewness. The model is saved only if it demonstrates improved predictability compared to the previously saved model. If not, that particular training iteration is discarded, and retraining starts with the previous best model using the next streaming CAN data. As we carefully selected the learning rate, decay rate, and momentum to mitigate overfitting, scenarios of overfitting were not observed, enabling us to retrain with the complete dataset.

The improvement in ID prediction accuracy for both the pre-trained and re-trained models, specifically for the model trained on low-frequency IDs, is illustrated in Figure 9.6. Notably, these low-frequency IDS exhibit a substantial increase in accuracy as a consequence of retraining with a larger dataset. This underscores the effectiveness of the retraining process in enhancing ID prediction accuracy, directly influencing the accuracy of anomaly detection. Further enhancements in these accuracies can be achieved by retraining the models with additional large benign datasets.



(a) Pre-trained model softmax probabilities (b) Re-trained model softmax probabilities

Figure 9.4: Softmax probability distribution change for ID 50C

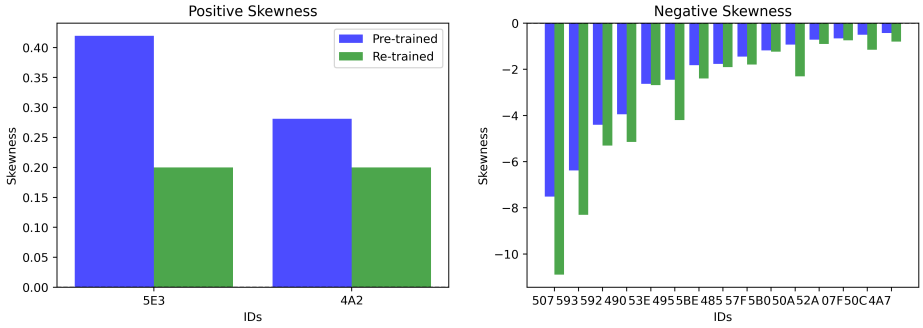


Figure 9.5: ID skewness change

9.5.3 Results and Discussion

This section discusses the detection performance during model deployment, focusing on both detection capability and detection latency.

Attack detection

During the deployment, while the IDS is monitoring the CAN streaming data, we executed various attacks and observed near real-time detection for these attacks. Some of these results can be observed in the demonstration video at https://www.youtube.com/watch?v=CufiACr2Zs8. However, this does not allow for the evaluation of the percentage accuracy of attack detection. Therefore, we evaluated the accuracy of attack detection using the attack dataset created and introduced in Chapter 8. As we evaluate the detection latency during real deployment, for accuracy evaluation, we conducted experiments on a MacBook M1 Pro using the same models integrated into the Raspberry Pi. The

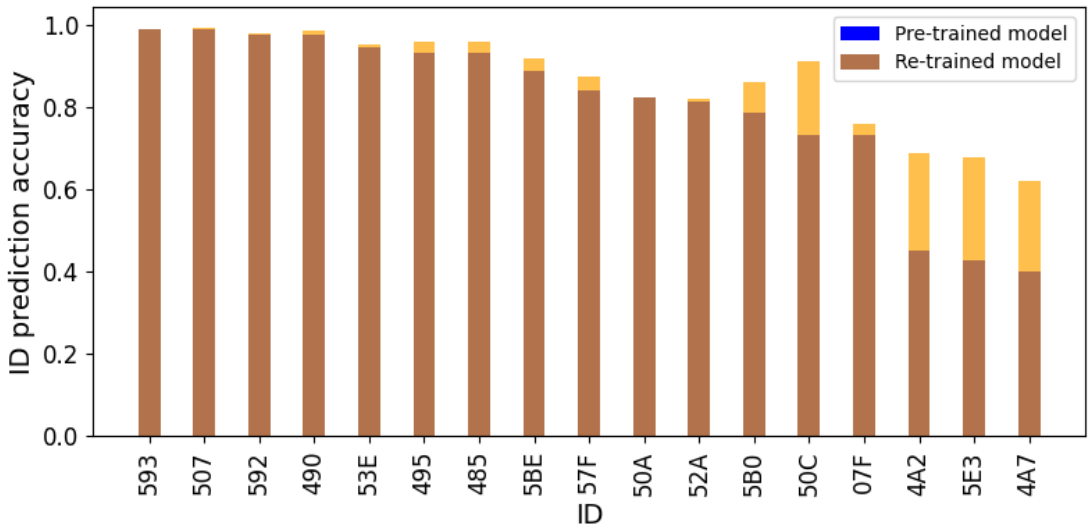


Figure 9.6: Model progress with the retraining

results of these evaluations are shown in Table 9.1, Table 9.2, Table 9.3 and Table 9.4. We also used the time-based model introduced in Chapter 5 for the model comparison.

Based on the observed results, the retrained CAN-ODTL model consistently outperformed the pre-trained CAN-ODTL model across all injection and masquerade attacks. Particularly for attacks targeting low-frequency IDs, the Latent AE model surpassed the performance of the CAN-ODTL model, as these IDs required more retraining to achieve high detection accuracy. Given its design to learn payload associations, Latent AE excelled in detecting all masquerade attacks compared to the CAN-ODTL model. The wiper warning attacks specifically targeted a low-frequency ID and involved minimal changes to the payload. Consequently, both the CAN-ODTL and Latent AE models struggled to achieve a higher detection rate for this attack. The parking brake attack only slightly altered the seventh byte of the payload. Consequently, Latent AE struggled to detect this attack with a higher detection rate. Conversely, as this attack targeted a medium-frequency ID, CAN-ODTL achieved a 100% detection rate. For the suspension attacks, the Latent AE model achieved 100% attack detection (TP). This occurred because Latent AE utilises the most recent payload for each ID. During a suspension attack, where the target ID does not appear, Latent AE uses the last transmitted ID payload as the most recent payload values for the targeted ID. Consequently, this creates an association mismatch with the associated variable throughout the attack period, resulting in the 100% attack detection rate. While the time-based model exhibited higher detection rates

Table 9.1: Comparison of Time-based, CAN-ODTL Pre-trained (CAN-ODTL PT), CAN-ODTL Re-trained (CAN-ODTL RT), and Latent AE IDS detection performance with CAN-MIRGU dataset injection attacks

Attack	Model	F1	TP	TN	FP	FN
Steering angle	Time-based	91.9%	100%	88.7%	11.2%	0.0%
	CAN-ODTL PT	99.4%	100%	99.1%	0.8%	0.0%
	CAN-ODTL RT	**99.8**%	100%	99.6%	0.1%	0.0%
	Latent AE	98.1%	98.6%	98.9%	1.1%	1.4%
Steering angle replay	Time-based	97.4%	100%	95.6%	4.3%	0.0%
	CAN-ODTL PT	99.2%	100%	98.4%	1.5%	0.0%
	CAN-ODTL RT	99.9%	100%	99.3%	0.7%	0.0%
	Latent AE	**100**%	100%	100%	0.0%	0.0%
Break warning	Time-based	94.2%	100%	91.4%	8.5%	0.0%
	CAN-ODTL PT	96.4%	100%	97.5%	2.5%	0.0%
	CAN-ODTL RT	**98.1**%	100%	98.7%	1.3%	0.0%
	Latent AE	96.2%	92.7%	99.2%	0.8%	7.3%
FCA warning	Time-based	99.2%	100%	98.9%	1.0%	0.0%
	CAN-ODTL PT	99.9%	100%	99.7%	0.2%	0.0%
	CAN-ODTL RT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	91.5%	91.2%	100%	0.0%	9.8%
EMS	Time-based	98.9%	100%	98.0%	1.9%	0.0%
	CAN-ODTL PT	99.9%	100%	99.2%	0.7%	0.0%
	CAN-ODTL RT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	99.9%	99.9%	100%	0.0%	0.1%
EMS replay	Time-based	91.1%	100%	88.8%	14.1%	0.0%
	CAN-ODTL PT	98.2%	100%	97.1%	2.8%	0.0%
	CAN-ODTL RT	**99.7**%	100%	99.3%	0.7%	0.0%
	Latent AE	99.5%	99.4%	100%	0.0%	0.6%
EMS replay long	Time-based	95.2%	100%	92.5%	7.4%	0.0%
	CAN-ODTL PT	98.0%	99.1%	97.8%	2.2%	0.9%
	CAN-ODTL RT	98.1%	99.3%	98.9%	1.1%	0.7%
	Latent AE	**98.6**%	100%	98.5%	1.5%	0.0%
Gear shifter 1	Time-based	99.4%	100%	98.7%	1.2%	0.0%
	CAN-ODTL PT	**100**%	100%	100%	0.0%	0.0%
	CAN-ODTL RT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	99.9%	100%	99.8%	0.2%	0.0%
Gear shifter 2	Time-based	93.3%	100%	89.4%	10.5%	0.0%
	CAN-ODTL PT	99.8%	100%	99.6%	0.3%	0.0%
	CAN-ODTL RT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	99.3%	100%	99.9%	0.1%	0.0%
Power steering	Time-based	98.4%	100%	97.9%	2.0%	0.0%
	CAN-ODTL PT	68.8%	34.2%	99.5%	0.4%	65.7%
	CAN-ODTL RT	92.1%	94.2%	99.9%	0.1%	5.8%
	Latent AE	**99.9**%	100%	99.9%	0.1%	0.0%
Max speedometer	Time-based	93.8%	99.8%	92.7%	7.2%	0.1%
	CAN-ODTL PT	98.5%	99.8%	97.7%	2.2%	0.1%
	CAN-ODTL RT	99.5%	99.9%	99.8%	0.2%	0.1%
	Latent AE	**100**%	100%	100%	0.0%	0.0%
Min speedometer	Time-based	98.1%	100%	96.0%	3.9%	0.0%
	CAN-ODTL PT	99.7%	100%	99.6%	0.4%	0.0%
	CAN-ODTL RT	**99.9**%	100%	99.9%	0.1%	0.0%
	Latent AE	**99.9**%	100%	99.8%	0.2%	0.0%
Parking break	Time-based	92.5%	100%	90.9%	9.0%	0.0%
	CAN-ODTL PT	99.8%	100%	99.8%	0.2%	0.0%
	CAN-ODTL RT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	93.1%	91.4%	100%	0.0%	8.6%

Table 9.2: Comparison of Time-based, CAN-ODTL Pre-trained (CAN-ODTL PT), CAN-ODTL Re-trained (CAN-ODTL RT), and Latent AE IDS detection performance with CAN-MIRGU dataset injection attacks

Attack	Model	F1	TP	TN	FP	FN
DoS	Time-based	**99.8**%	100.0%	99.2%	0.2%	0.0%
	CAN-ODTL PT	99.7%	100%	99.2%	0.8%	0.0%
	CAN-ODTL RT	**99.8**%	100%	99.7%	0.3%	0.0%
	Latent AE	**99.8**%	100%	99.5%	0.5%	0.0%
Fuzzing random ID	Time-based	**99.9**%	100%	99.1%	0.1%	0.0%
	CAN-ODTL PT	99.8%	100%	99.6%	0.4%	0.0%
	CAN-ODTL RT	**99.9**%	100%	99.8%	0.2%	0.0%
	Latent AE	99.7%	100%	99.7%	0.3%	0.0%
Fuzzing Valid ID	Time-based	84.7%	100%	84.3%	15.6%	0.0%
	CAN-ODTL PT	96.1%	99.3%	96.2%	3.7%	0.6%
	CAN-ODTL RT	**99.8**%	100%	99.9%	0.1%	0.0%
	Latent AE	99.5%	100%	99.3%	0.7%	0.0%
Drive mode changing	Time-based	**99.7**%	100%	99.4%	0.5%	0.0%
	CAN-ODTL PT	88.1%	69.8%	99.1%	0.9%	30.1%
	CAN-ODTL RT	93.3%	84.1%	99.6%	0.4%	15.9%
	Latent AE	99.5%	100%	99.3%	0.7%	0.0%
Door open warning	Time-based	89.5%	75.9%	99.2%	0.7%	24.0%
	CAN-ODTL PT	88.0%	69.8%	99.8%	0.1%	30.1%
	CAN-ODTL RT	94.5%	96.4%	100%	0.0%	3.6%
	Latent AE	**99.8**%	99.6%	100%	0.0%	0.4%
Wiper warning	Time-based	65.5%	34.2%	93.4%	6.5%	65.7%
	CAN-ODTL PT	84.1%	74.1%	92.7%	7.2%	25.8%
	CAN-ODTL RT	89.5%	86.2%	94.8%	5.2%	13.8%
	Latent AE	**91.8**%	92.6%	96.2%	3.8%	7.4%

Table 9.3: Comparison of Time-based, CAN-ODTL Pre-trained (CAN-ODTL PT), CAN-ODTL Re-trained (CAN-ODTL RT), and Latent AE IDS detection performance with CAN-MIRGU dataset masquerade attacks

Attack	Model	F1	TP	TN	FP	FN
Steering angle masquerade	Time-based	38.1%	0.9%	88.7%	11.2%	99.0%
	CAN-ODTL PT	62.1%	25.4%	99.1%	0.8%	75.5%
	CAN-ODTL RT	91.2%	71.1%	99.6%	0.4%	29.9%
	Latent AE	**99.2**%	99.9%	99.9%	0.1%	0.1%
Break warning masquerade	Time-based	48.3%	13.2%	91.4%	8.5%	86.7%
	CAN-ODTL PT	91.2%	100%	84.3%	15.6%	0.0%
	CAN-ODTL RT	97.2%	100%	97.9%	2.1%	0.0%
	Latent AE	**99.7**%	99.9%	99.9%	0.1%	0.1%
Min speedometer masquerade	Time-based	44.3%	13.6%	96.0%	3.9%	86.3%
	CAN-ODTL PT	99.1%	98.2%	99.6%	0.3%	1.7%
	CAN-ODTL RT	99.3%	98.9%	99.8%	0.2%	1.1%
	Latent AE	**99.5**%	99.3%	99.9%	0.1%	0.7%
Wiper warning masquerade	Time-based	54.6%	17.4%	93.4%	6.5%	82.5%
	CAN-ODTL PT	88.4%	76.1%	99.1%	0.9%	24.9%
	CAN-ODTL RT	94.1%	89.7%	99.6%	0.4%	10.3%
	Latent AE	**99.1**%	99.4%	99.5%	0.5%	0.6%
Break and fog light masquerade	Time-based	50.1%	11.8%	90.3%	9.6%	88.1%
	CAN-ODTL PT	81.3%	71.6%	96.1%	3.9%	28.4%
	CAN-ODTL RT	92.3%	90.2%	99.6%	0.4%	9.8%
	Latent AE	**98.3**%	98.8%	99.1%	0.9%	1.2%

Table 9.4: Comparison of Time-based, CAN-ODTL Pre-trained (CAN-ODTL PT), CAN-ODTL Re-trained (CAN-ODTL RT), and Latent AE IDS detection performance with CAN-MIRGU dataset suspension attacks

Attack	Model	F1	TP	TN	FP	FN
ID 160 suspension	Time-based	98.7%	100%	98.4%	1.6%	0.0%
	CAN-ODTL PT	99.7%	100%	99.1%	0.1%	0.0%
	CAN-ODTL RT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	**100**%	100%	99.8%	0.2%	0.0%
ID 371 suspension	Time-based	96.3%	100%	93.2%	6.8%	0.0%
	CAN-ODTL PT	99.3%	100%	99.6%	0.4%	0.0%
	CAN-ODTL RT	**100**%	100%	99.9%	0.1%	0.0%
	Latent AE	**100**%	100%	99.9%	0.1%	0.0%
ID 386 suspension	Time-based	98.3%	100%	97.1%	2.9%	0.0%
	CAN-ODTL PT	**100**%	100%	100%	0.0%	0.0%
	CAN-ODTL RT	**100**%	100%	100%	0.0%	0.0%
	Latent AE	99.9%	100%	99.9%	0.1%	0.0%
ID 541 suspension	Time-based	90.2%	100%	89.4%	9.6%	0.0%
	CAN-ODTL PT	87.1%	67.6%	99.9%	0.1%	32.4%
	CAN-ODTL RT	93.2%	90.3%	98.1%	1.9%	9.7%
	Latent AE	**98.7**%	100%	99.1%	0.4%	0.0%
ID 07F suspension	Time-based	85.6%	100%	89.2%	10.8%	0.0%
	CAN-ODTL PT	85.6%	78.1%	92.1%	7.9%	21.9%
	CAN-ODTL RT	91.0%	86.2%	97.3%	2.7%	13.8%
	Latent AE	**98.8**%	100%	99.3%	0.7%	0.0%

for injection attacks, it is susceptible to false positives and lacks the ability to consider the context of ID transmission. In summary, the integration of CAN-ODTL and Latent AE models proved to enhance detection capabilities across all types of attacks.

Overhead analysis

CAN-ODTL comprises three models, each of which can undergo retraining for a specific duration. Among these models, model-1, which encompasses lower frequency IDs, requires retraining with a more extensive dataset compared to model-2 and model-3, which handle medium and high-frequency IDs, respectively. The Table 9.5 illustrates the average retraining time per batch, as well as the CPU and RAM utilisation of the Raspberry Pi across the retraining sessions for all three models. Despite being lightweight versions, model-2 and model-3 exhibit higher CPU and RAM consumption during retraining compared to model-1. This disparity can be attributed to the frequently appearing IDs associated with model-2 and model-3. During model retraining, inference must be paused until retraining is completed, as running both processes concurrently leads to increased inference time, which is not suitable to making near real-time predictions.

The average inference time overhead per CAN frame for CAN-ODTL and Latent AE models is presented in Table 9.6. When running both models in parallel, the inference

Table 9.5: CAN-ODTL models average retraining overhead on Raspberry Pi

Model	Retraining time (s)	CPU usage (%)	RAM usage (%)
model-1	0.13	36	123
model-2	0.10	41	128
model-3	0.08	46	144

Table 9.6: CAN-ODTL and Latent AE average inference overhead on Raspberry Pi

Model	TFLite Model size (KB)	Inference time(ms)
CAN-ODTL	29	0.28
Latent AE	22	0.32
CAN-ODTL + Latent AE	51	0.44

process takes approximately 0.44ms to make a prediction, whereas the average frame arrival rate is 0.5ms. This results in 78% CPU usage and 86MB RAM consumption. Since this approach uses time windows to make predictions instead of predicting for each frame, a 25ms window size ise selected as the optimal size to minimize false positives. During deployment, attack detection alerts are promptly generated upon the execution of attack scripts, thereby achieving near real-time detection.

9.6 Limitations

Despite the promising results achieved with the model deployment, there are several limitations to consider. The model limitations and assumptions discussed in previous chapters, such as overlooking important variables and challenges in observing minimum and maximum values in the Latent AE model, are applicable to the improved model versions used during deployment. The vehicle used for the deployment experiments only included 56 CAN IDs. However, vehicles with a higher number of IDs may increase the model complexity of both CAN-ODTL and Latent AE models, as the model architecture depends on the number of CAN IDs and payload variables. In such cases, running both models on the same Raspberry Pi may not be feasible due to limited computing power. One potential solution is to deploy two models on two Raspberry Pis for ID and payload monitoring. Due to the limited time received for model deployment experiments, model retraining relied on virtual CAN bus replaying collected CAN logs without prolonged retraining of CAN-ODTL during deployment. Additionally, these models were only tested on one vehicle in a controlled environment and require extensive experiments with multiple vehicles and conditions to evaluate the effectiveness of the proposed models under various driving conditions.

9.7 Conclusion

Previous research efforts in CAN IDS deployment primarily emphasized model development and enhancing attack detection capabilities. Owing to the numerous challenges associated with executing attacks on an actual vehicle, these studies did not focus on deploying the model onto a real vehicle and assessing its performance in real-world conditions.

Hence, this chapter centres on the deployment and evaluation of IDS on an actual vehicle. The experiments revealed that even a minimal detection latency is unacceptable for an IDS intended for deployment in a real vehicle. Consequently, we refined the proposed CAN ID and Payload-based IDSs to achieve near-real-time detection. Experiments involving the retraining of CAN-ODTL demonstrated a significant enhancement in ID prediction and, consequently, attack detection through retraining with an extensive benign dataset. The integration of both ID-based and payload-based models for monitoring CAN data resulted in an overall improvement in attack detection. Performance experiments conducted in real-world settings indicated that the proposed models are well-suited for deployment in a real vehicle, enabling the detection of a wide variety of attacks with near-real-time capabilities.

Chapter 10

Conclusion

This chapter summarizes the key outcomes of the preceding chapters and discusses the limitations and future directions of this body of work.

10.1 Summary

This work contributes to the state-of-the-art advancement in developing practically deployable CAN IDS to improve the security of IVNs. The method proposed in this thesis addresses several key challenges in IVN security.

The development of CAN IDSs faces several challenges, including the limited availability of attack datasets, constrained computing power in the In-Vehicle IVN environment, the absence of CAN data specifications, and the necessity for near-real-time detection in high-speed CAN data. Typically, attackers employ injection, masquerade, and suspension attacks to compromise various functionalities of a vehicle. In response to these attacks, CAN frame fields, such as ID and payload, manifest anomalous behaviours that can be leveraged for attack detection on the CAN bus. To effectively detect injection attacks, as detailed in Chapter 5, we utilised CAN ID sequences and ID inter-arrival change patterns. A GRU-based model was employed to learn the ID sequences, while a time-based model was utilised to capture the ID inter-arrival patterns. This IDS demonstrated effectiveness against injection and masquerade attacks that subtly altered ID sequences. However, not all injection attacks may alter the ID sequences sufficiently to surpass the defined thresholds. Future work should investigate the impact of varying injection frequencies on high, medium, and low-frequency IDs, and how to optimize thresholds

for improved attack detection. In this model, anomalies are assumed to be attacks, but in practice, not all anomalies are attacks. Further exploration is needed to distinguish between anomalies and actual attacks. A notable limitation of the proposed lightweight GRU-based model is its reliance on a large dataset to mitigate unseen sequences and enhance attack detection capabilities.

To overcome this challenge, Chapter 6 introduces CAN-ODTL to retrain the classification layer of the GRU-based model using streaming CAN data. The use of an optimized data-processing algorithm enables efficient retraining, particularly in a resource-constrained Raspberry Pi environment. This approach proves effective in enhancing attack detection by retraining the model with a substantial benign dataset encompassing diverse driving behaviours. However, further experiments involving poisoning the streaming data through label flipping reveal the vulnerability of this approach to data poisoning attacks. To mitigate the potential impact of such attacks, we propose an effective technique based on the Mahalanobis distance to sanitize the data before retraining. This technique demonstrates its efficacy in detecting even a small percentage of data poisoning instances, ensuring data integrity during the retraining process. This approach concentrated solely on label flipping and did not address advanced data poisoning strategies. It is important to evaluate the effectiveness of the proposed method against such sophisticated attacks in future work.

Given that CAN-ODTL exclusively relies on the CAN ID field, detecting attacks solely manipulating the CAN payload field, such as sophisticated masquerade attacks, poses a challenge. Consequently, the need arises for an IDS that leverages information from the CAN payload field. In Chapter 7, a novel AE-based IDS is introduced to address this gap. Payload-based IDSs encounter a significant challenge due to the lack of prior knowledge about the CAN data specification. To address this, we adopt a strategy of utilising raw payload values to learn associations between variables. To reduce the dimensionality of the payload field, Cramér's $\hat{V}$ statistics are employed, proving more effective than Pearson correlation for CAN payload data. Nevertheless, the Vanilla AE model exhibits overgeneralization issues, particularly for variables with limited unique values. To address this, an enhanced AE architecture is proposed by introducing a small AE into the latent space. Integration of this model with the GRU-based model demonstrates improved detection rates for both injection and masquerade attacks. However, the Latent AE model has certain limitations. Not considering the CAN specification may cause it to overlook crucial variables, and determining true minimum and maximum values for each variable proves challenging, potentially leading to false positives or false negatives. Implementing

a payload field classification algorithm and subsequently using precise payload features with the Latent AE will enhance both attack detection and detection latency, thereby reducing its limitations.

During this research project, we realise that there is a discernible need for a comprehensive CAN bus attack dataset featuring realistic and verified attacks. The existing datasets, with the ROAD dataset being the primary resource, partly address this need. However, the ROAD dataset also comes with notable limitations, such as the absence of attack data collection during active road driving. To fill this void, we present CAN-MIRGU, a novel CAN bus attack dataset gathered from a moving vehicle. This dataset is meticulously curated to overcome the limitations inherent in publicly available datasets. It encompasses approximately 17 hours of benign data and 3 hours of attack data, encompassing 36 distinct attacks. This inclusive dataset aims to facilitate more robust comparisons and validations of various CAN IDS solutions.

Ultimately, we executed the deployment of CAN-ODTL and Latent AE on an actual vehicle to assess their effectiveness in real-world scenarios. These models were seamlessly integrated into a Raspberry Pi device and connected to the CAN bus through the central gateway. Prior to deployment, the CAN-ODTL model underwent retraining with an extensive dataset, resulting in improved accuracy. Significantly, achieving a detection latency faster than the CAN bus data transmission speed was paramount. This necessitated further enhancements to the CAN-ODTL and Latent AE models to enable near-real-time detection. Experiments involving realistic attacks on a moving vehicle revealed that both models could effectively operate in real-world settings, detecting attacks with near-real-time capabilities. However, the deployment experiments were conducted within a limited timeframe, highlighting the need for more extensive experiments with a variety of vehicles to thoroughly evaluate the models under different conditions.

10.2 Objectives Revisited

In this section, We revisit the research objectives set out in Section 1.2 and summarize how each objective has been addressed throughout this doctoral research.

- **Conduct a thorough examination of IVN attack and countermeasures:** This objective has been addressed in Chapter 2 and Chapter 3, laying the foundation for the proposed CAN ID-based and payload-based IDSs discussed in subsequent chapters. These solutions aim to tackle the existing challenges in the field.

- **Develop a lightweight, AI-based IDS based on CAN IDs:** This objective has been addressed in Chapter 5 and Chapter 6. The outcomes demonstrate that the ID-based model can effectively detect injection and masquerade attacks in a resource-constrained edge device.
- **Developing a lightweight CAN payload-based IDS:** This objective has been addressed in Chapter 7 through the implementation of a novel AE-based IDS that leverages the CAN payload data.
- **Generate a comprehensive CAN bus attack dataset:** This objective has been addressed in Chapter 8 through the introduction of CAN-MIRGU, a comprehensive CAN bus attack dataset collected from moving vehicles for IDS evaluation. This dataset comprises 17 hours of benign data and 3 hours of attack data, including 36 unique attacks.
- **Model deployment:** This objective has been addressed in Chapter 9 by deploying the enhanced CAN-ODTL and Latent AE models in a real vehicle. Integration into Raspberry Pi devices was performed, and the models were evaluated with real-world attacks.

10.3 Future Directions

Throughout this work, we have identified several potential improvements and future directions.

10.3.1 Streaming learning

Our experiments with CAN-ODTL have demonstrated the effectiveness of streaming learning in improving attack detection. However, we were unable to evaluate the streaming learning's performance over an extended period to determine whether its short-term effectiveness can be maintained in the long run. This limitation was due to restricted access to the vehicle and the conclusion of the project. Future research should aim to retrain CAN-ODTL model under various driving conditions while the vehicle is operational, assessing and enhancing the system's long-term performance over extended periods, such as a year or more. Furthermore, automating the entire streaming learning pipeline represents another essential research area for exploration. Preliminary experiments on training Latent AE with various dataset sizes indicated that Latent AE can also benefit from a large dataset for training. Therefore, integrating streaming learning

into the model can improve its ability to learn all possible variable associations. However, addressing the issue of overgeneralization during the retraining procedure is crucial. To this end, the latent space autoencoder introduced in Chapter 7 can be further improved.

10.3.2 Testing on other vehicles

Due to time and resource constraints, we were only able to collect benign and attack data from a single vehicle. Consequently, the deployment experiments outlined in Chapter 9 were conducted solely on this vehicle, which may limit the generalizability of our findings. This limitation arises because CAN data characteristics, such as the number of CAN IDs, message inter-arrival times, and specific CAN protocol configurations and payload sensor values, can vary significantly across different car makes and models. Therefore, future work should focus on generating more benign and attack data from multiple vehicles to assess the generalizability of the CAN-ODTL and Latent AE models. Furthermore, despite the vehicle used being equipped with fully autonomous driving capabilities, none of the experiments were conducted while the vehicle was in autonomous mode. Exploring the performance of these models on autonomous vehicles represents a promising avenue for future research.

10.3.3 Distinguish cyberattacks and benign anomalies

Distinguishing between benign anomalies and cyberattacks is challenging. Since the proposed models are based on anomaly detection techniques, we assume anomalies may indicate cyberattacks. However, rare benign driving activities or technical issues can also generate anomalies. To reduce the likelihood of false positives, we used a large training dataset and monitored the count of weak anomalies over a window to declare an attack. Despite this, benign anomalies might still be flagged as attacks. Future research should focus on developing methods to distinguish between benign anomalies and cyberattacks, ensuring alerts are only triggered for real cyberattacks.

10.3.4 Integrate the models directly into ECUs

We have integrated CAN-ODTL and Latent AE into the Raspberry Pi device and then connected it to the CAN bus. However, deploying these models directly into an Electronic Control Unit (ECU) would be more efficient. However, this poses challenges due to the limited memory and computational resources available in ECUs. Techniques employed in TinyML can be explored to address these challenges. Additionally, retraining CAN-ODTL in ECUs would be more challenging, but it is worth exploring as it would enable

the direct embedding of these models into the hardware, further reducing inference time.

10.3.5 Countermeasures against cyberattacks

The deployed IDSs demonstrated their effectiveness in detecting various attacks. Additionally, attacks on the vehicle revealed that their impact is immediate. Therefore, implementing effective countermeasures based on the attack detection is crucial. One possible approach could involve deactivating targeted systems soon after detecting the attack to minimize the impact. However, this may not be practical for safety-critical systems. Therefore, it is important to explore potential countermeasures, possibly in collaboration with original equipment manufacturers, as these countermeasures should be automated to take immediate actions.

10.3.6 Model tampering attacks

We explored the effects of data poisoning attacks on model training and introduced a data poisoning defense technique in Section 6.5. However, due to time constraints, we could not test this defense procedure across different driving conditions, varying percentages of data poisoning, and more sophisticated data poisoning strategies. Therefore, evaluating its efficacy in diverse scenarios should be a focus of future work. Additionally, it is important to acknowledge the potential threat of model tampering attacks by adversaries, which could compromise the performance of AI-based models. This presents another area worth investigating, particularly given the susceptibility of AI models to adversarial attacks. In this regard, the utilisation of longer attack logs introduced in Chapter 8 can be instrumental.

Bibliography

[1] Aliwa E, Rana O, Perera C, Burnap P. Cyberattacks and countermeasures for in-vehicle networks. ACM Computing Surveys (CSUR). 2021;54(1):1-37.

[2] Ag V, Wolfsburg G. Data exchange on the CAN-bus I basic. VAG SSP 238, 2001. Order;(140.2810):57-20.

[3] Song HM, Kim HR, Kim HK. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In: 2016 international conference on information networking (ICOIN). IEEE; 2016. p. 63-8.

[4] Charette RN. This Car Runs on Code; 2009. Retrieved July 2021 from https://spectrum.ieee.org/transportation/systems/this-car-runs-on-code.

[5] Boumiza S, Braham R. Intrusion threats and security solutions for autonomous vehicle networks. In: 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA). Hammamet, Tunisia: IEEE; 2017. p. 120-7.

[6] Al-Jarrah OY, Maple C, Dianati M, Oxtoby D, Mouzakitis A. Intrusion detection systems for intra-vehicle networks: A review. IEEE Access. 2019;7:21266-89.

[7] Rajapaksha S, Kalutarage H, Al-Kadri MO, Petrovski A, Madzudzo G. Improving In-vehicle Networks Intrusion Detection Using On-Device Transfer Learning. In: Symposium on Vehicles Security and Privacy (VehicleSec) 2023;. .

[8] Koscher K, Czeskis A, Roesner F, Patel S, Kohno T, Checkoway S, et al. Experimental security analysis of a modern automobile. In: 2010 IEEE symposium on security and privacy. IEEE; 2010. p. 447-62.

[9] Miller C, Valasek C. Can message injection; 2016. Retrieved July 2021 from http://illmatics.com/can%20message%20injection.pdf.

[10] Cai Z, Wang A, Zhang W, Gruffke M, Schweppe H. 0-days & mitigations: Roadways to exploit and secure connected bmw cars. Black Hat USA. 2019;2019:39.

[11] Nie S, Liu L, Du Y. Free-fall: Hacking tesla from wireless to can bus. Briefing, Black Hat USA. 2017;25(1):16.

[12] UNECE. UN Regulation No. 155 - Cyber security and cyber security management system; 2021. Retrieved May 2024 from https://unece.org/transport/documents/2021/03/standards/un-regulation-no-155-cyber-security-and-cyber-security.

[13] UNECE. UN Regulation No. 156 - Software update and software update management system; 2021. Retrieved May 2024 from https://unece.org/transport/documents/2021/03/standards/un-regulation-no-156-software-update-and-software-update.

[14] ISO. Road vehicles — Cybersecurity engineering; 2021. Retrieved May 2024 from https://www.iso.org/obp/ui/en/#iso:std:iso-sae:21434:ed-1:v1:en.

[15] Engstler M. Heavy On Connectivity, Light On Security: The Challenges Of Vehicle Manufacturers;. Available from: https://www.forbes.com/sites/forbestechcouncil/2021/01/15/heavy-on-connectivity-light-on-security-the-challenges-of-vehicle-manufacturers/?sh=31913c607fc7.

[16] Lambert F. Tesla is challenging hackers to crack its car, and it is putting 1USD million on the line;. Accessed: 17.01.2024. https://electrek.co/2020/01/10/tesla-hacking-challenge/.

[17] Rajapaksha S, Kalutarage H, Al-Kadri MO, Petrovski A, Madzudzo G, Cheah M. AI-Based Intrusion Detection Systems for In-Vehicle Networks: A Survey. ACM Comput Surv. 2023 feb;55(11). Available from: https://doi.org/10.1145/3570954.

[18] Dutta L, Bharali S. Tinyml meets iot: A comprehensive survey. Internet of Things. 2021;16:100461.

[19] Lokman SF, Othman AT, Abu-Bakar MH. Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review. EURASIP Journal on Wireless Communications and Networking. 2019;2019(1):1-17.

[20] Kumar BV, Ramesh J. Automotive in vehicle network protocols. In: 2014 International Conference on Computer Communication and Informatics. IEEE; 2014. p. 1-5.

[21] Kishikawa T, Hirano R, Ujiie Y, Haga T, Matsushima H, Fujimura K, et al. Vulnerability of FlexRay and Countermeasures. SAE International Journal of Transportation Cybersecurity and Privacy. 2019;2(11-02-01-0002):21-33.

[22] Huang T, Zhou J, Wang Y, Cheng A. On the security of in-vehicle hybrid network: Status and challenges. In: Information Security Practice and Experience: 13th International Conference, ISPEC 2017, Melbourne, VIC, Australia, December 13–15, 2017, Proceedings 13. Springer; 2017. p. 621-37.

[23] Avatefipour O, Malik H. State-of-the-art survey on in-vehicle network communication (CAN-Bus) security and vulnerabilities. IJCSN. 2017;6(6):720-7.

[24] Liu J, Zhang S, Sun W, Shi Y. In-vehicle network attacks and countermeasures: Challenges and future directions. IEEE Network. 2017;31(5):50-8.

[25] El-Rewini Z, Sadatsharan K, Selvaraj DF, Plathottam SJ, Ranganathan P. Cybersecurity challenges in vehicular communications. Vehicular Communications. 2020;23:100214.

[26] Rajapaksha S, Kalutarage H, Al-Kadri MO, Madzudzo G, Petrovski AV. 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). vol. 700. IEEE; 2022. p. 309-30.

[27] Avatefipour O, Al-Sumaiti AS, El-Sherbeeny AM, Awwad EM, Elmeligy MA, Mohamed MA, et al. An intelligent secured framework for cyberattack detection in electric vehicles' CAN bus using machine learning. IEEE Access. 2019;7:127580-92.

[28] Zhou A, Li Z, Shen Y. Anomaly detection of CAN bus messages using a deep neural network for autonomous vehicles. Applied Sciences. 2019;9(15):3174.

[29] Markovitz M, Wool A. Field classification, modeling and anomaly detection in unknown CAN bus networks. Vehicular Communications. 2017;9:43-52.

[30] Verma ME, Iannacone MD, Bridges RA, Hollifield SC, Kay B, Combs FL. ROAD: The Real ORNL Automotive Dynamometer Controller Area Network Intrusion Detection Dataset (with a comprehensive CAN IDS dataset survey & guide). arXiv preprint arXiv:201214600. 2020.

[31] Miller C. Lessons learned from hacking a car. IEEE Design & Test. 2019;36(6):7-9.

[32] Rajapaksha S, Kalutarage H, Al-Kadri MO, Petrovski A, Madzudzo G. Beyond vanilla: Improved autoencoder-based ensemble in-vehicle intrusion detection system. Journal of information security and applications. 2023;77:103570.

[33] Miller C, Valasek C. Remote exploitation of an unaltered passenger vehicle. Black Hat USA. 2015;2015(S 91).

[34] Rajapaksha S, Kalutarage H, Al-Kadri MO, Petrovski A, Madzudzo G, Cheah M. AI-Based Intrusion Detection Systems for In-Vehicle Networks: A Survey. ACM Comput Surv. 2023 feb;55(11). Available from: https://doi.org/10.1145/3570954.

[35] Fakhfakh F, Tounsi M, Mosbah M. Cybersecurity attacks on CAN bus based vehicles: a review and open challenges. Library hi tech. 2022;40(5):1179-203.

[36] Petit J, Stottelaar B, Feiri M, Kargl F. Remote attacks on automated vehicles sensors: Experiments on camera and lidar. Black Hat Europe. 2015;11(2015):995.

[37] Dürrwang J, Braun J, Rumez M, Kriesten R, Pretschner A. Enhancement of automotive penetration testing with threat analyses results. SAE International Journal of Transportation Cybersecurity and Privacy. 2018;1(11-01-02-0005):91-112.

[38] Iehira K, Inoue H, Ishida K. Spoofing attack using bus-off attacks against a specific ECU of the CAN bus. In: 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC). IEEE; 2018. p. 1-4.

[39] Fan J, Yan Q, Li M, Qu G, Xiao Y. A Survey on Data Poisoning Attacks and Defenses. In: 2022 7th IEEE International Conference on Data Science in Cyberspace (DSC). IEEE; 2022. p. 48-55.

[40] Wang Z, Ma J, Wang X, Hu J, Qin Z, Ren K. Threats to Training: A Survey of Poisoning Attacks and Defenses on Machine Learning Systems. ACM Comput Surv. 2022 dec;55(7). Available from: https://doi.org/10.1145/3538707.

[41] Haykin S. Neural networks: a comprehensive foundation. Prentice Hall PTR; 1998.

[42] Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks. 1994;5(2):157-66.

[43] Salehinejad H, Sankar S, Barfett J, Colak E, Valaee S. Recent advances in recurrent neural networks. arXiv preprint arXiv:180101078. 2017.

[44] Guo Y, Shi H, Kumar A, Grauman K, Rosing T, Feris R. Spottune: transfer learning through adaptive fine-tuning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition; 2019. p. 4805-14.

[45] Gong D, Liu L, Le V, Saha B, Mansour MR, Venkatesh S, et al. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision; 2019. p. 1705-14.

[46] Denouden T, Salay R, Czarnecki K, Abdelzad V, Phan B, Vernekar S. Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance. arXiv preprint arXiv:181202765. 2018.

[47] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Advances in neural information processing systems. 2017;30.

[48] Tay Y, Dehghani M, Bahri D, Metzler D. Efficient Transformers: A Survey; 2022.

[49] Choukroun Y, Kravchik E, Yang F, Kisilev P. Low-bit quantization of neural networks for efficient inference. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). IEEE; 2019. p. 3009-18.

[50] Bai H, Hou L, Shang L, Jiang X, King I, Lyu MR. Towards efficient post-training quantization of pre-trained language models. Advances in Neural Information Processing Systems. 2022;35:1405-18.

[51] Ran X, Xi Y, Lu Y, Wang X, Lu Z. Comprehensive survey on hierarchical clustering algorithms and the recent developments. Artificial Intelligence Review. 2023;56(8):8219-64.

[52] Ackerman M, Ben-David S. A characterization of linkage-based hierarchical clustering. The Journal of Machine Learning Research. 2016;17(1):8182-98.

[53] Chai T, Draxler RR. Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. Geoscientific model development. 2014;7(3):1247-50.

[54] Narayanan SN, Mittal S, Joshi A. OBD_SecureAlert: An anomaly detection system for vehicles. In: 2016 IEEE International Conference on Smart Computing (SMARTCOMP). IEEE; 2016. p. 1-6.

[55] Bella G, Biondi P, Costantino G, Matteucci I. Toucan: A protocol to secure controller area network. In: Proceedings of the ACM Workshop on Automotive Cybersecurity; 2019. p. 3-8.

[56] Liberati A, Altman DG, Tetzlaff J, Mulrow C, Gøtzsche PC, Ioannidis JP, et al. The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions: explanation and elaboration. Journal of clinical epidemiology. 2009;62(10):e1-e34.

[57] Wohlin C. 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; 2014. p. 1-10.

[58] Kalutarage HK, Al-Kadri MO, Cheah M, Madzudzo G. Context-aware Anomaly Detector for Monitoring Cyber Attacks on Automotive CAN Bus. In: ACM Computer Science in Cars Symposium. New York, NY, USA: Association for Computing Machinery; 2019. p. 1-8. Available from: https://doi.org/10.1145/3359999.3360496.

[59] Marchetti M, Stabili D. Anomaly detection of CAN bus messages through analysis of ID sequences. In: 2017 IEEE Intelligent Vehicles Symposium (IV). IEEE; 2017. p. 1577-83.

[60] Wang Y, Chia DWM, Ha Y. Vulnerability of Deep Learning Model based Anomaly Detection in Vehicle Network. In: 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS). IEEE; 2020. p. 293-6.

[61] Liu H, Lang B. Machine learning and deep learning methods for intrusion detection systems: A survey. applied sciences. 2019;9(20):4396.

[62] Grekousis G. Artificial neural networks and deep learning in urban geography: A systematic review and meta-analysis. Computers, Environment and Urban Systems. 2019;74:244-56.

[63] Kang MJ, Kang JW. Intrusion detection system using deep neural network for in-vehicle network security. PloS one. 2016;11(6):e0155781.

[64] Desta AK, Ohira S, Arai I, Fujikawa K. Mlids: Handling raw high-dimensional can bus data using long short-term memory networks for intrusion detection in in-vehicle networks. In: 2020 30th International Telecommunication Networks and Applications Conference (ITNAC). IEEE; 2020. p. 1-7.

[65] Gao S, Zhang L, He L, Deng X, Yin H, Zhang H. Attack Detection for Intelligent Vehicles via CAN-Bus: A Lightweight Image Network Approach. IEEE Transactions on Vehicular Technology. 2023.

[66] Shahriar MH, Xiao Y, Moriano P, Lou W, Hou YT. CANShield: Deep Learning-Based Intrusion Detection Framework for Controller Area Networks at the Signal-Level. IEEE Internet of Things Journal. 2023.

[67] Seo E, Song HM, Kim HK. Gids: Gan based intrusion detection system for in-vehicle network. In: 2018 16th Annual Conference on Privacy, Security and Trust (PST). IEEE; 2018. p. 1-6.

[68] Levi M, Allouche Y, Kontorovich A. Advanced analytics for connected car cybersecurity. In: 2018 IEEE 87th Vehicular Technology Conference (VTC Spring). IEEE; 2018. p. 1-7.

[69] Tariq S, Lee S, Kim HK, Woo SS. CAN-ADF: The controller area network attack detection framework. Computers & Security. 2020;94:101857.

[70] Hanselmann M, Strauss T, Dormann K, Ulmer H. CANet: An unsupervised intrusion detection system for high dimensional CAN bus data. IEEE Access. 2020;8:58194-205.

[71] Cho KT, Shin KG. Error handling of in-vehicle networks makes them vulnerable. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: Association for Computing Machinery; 2016. p. 1044-55. Available from: https://doi.org/10.1145/2976749.2978302.

[72] Hacking, Lab CR. Car-Hacking Dataset for the intrusion detection; 2020. Retrieved August 2021 from https://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset.

[73] Chen M, Zhao Q, Jiang Z, Xu R. Intrusion Detection for in-vehicle CAN Networks Based on Auxiliary Classifier GANs. In: 2021 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS). IEEE; 2021. p. 186-91.

[74] Kavousi-Fard A, Dabbaghjamanesh M, Jin T, Su W, Roustaei M. An evolutionary deep learning-based anomaly detection model for securing vehicles. IEEE Transactions on Intelligent Transportation Systems. 2020;22(7):4478-86.

[75] Lee H, Jeong SH, Kim HK. 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). vol. 00; 2017. p. 57-5709.

[76] Al-Saud M, Eltamaly AM, Mohamed MA, Kavousi-Fard A. An intelligent data-driven model to secure intravehicle communications based on machine learning. IEEE Transactions on Industrial Electronics. 2019;67(6):5112-9.

[77] Rumez M, Lin J, Fuchß T, Kriesten R, Sax E. Anomaly Detection for Automotive Diagnostic Applications Based on N-Grams. In: 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE; 2020. p. 1423-9.

[78] Shi D, Xu M, Wu T, Kou L. Intrusion Detecting System Based on Temporal Convolutional Network for In-Vehicle CAN Networks. Mobile Information Systems. 2021;2021.

[79] Nam M, Park S, Kim DS. Intrusion detection method using bi-directional GPT for in-vehicle controller area networks. IEEE Access. 2021;9:124931-44.

[80] Baldini G. Intrusion detection systems in in-vehicle networks based on bag-of-words. In: 2021 5th Cyber Security in Networking Conference (CSNet). Abu Dhabi, United Arab Emirates: IEEE; 2021. p. 41-8.

[81] Alkhatib N, Mushtaq M, Ghauch H, Danger JL. CAN-BERT do it? Controller Area Network Intrusion Detection System based on BERT Language Model. In: 2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA). IEEE; 2022. p. 1-8.

[82] Desta AK, Ohira S, Arai I, Fujikawa K. ID sequence analysis for intrusion detection in the can bus using long short term memory networks. In: 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). IEEE; 2020. p. 1-6.

[83] Sharmin S, Mansor H. Intrusion Detection on the In-Vehicle Network Using Machine Learning. In: 2021 3rd International Cyber Resilience Conference (CRC). IEEE; 2021. p. 1-6.

[84] Kuwahara T, Baba Y, Kashima H, Kishikawa T, Tsurumi J, Haga T, et al. Supervised and unsupervised intrusion detection based on CAN message frequencies for in-vehicle network. Journal of Information Processing. 2018;26:306-13.

[85] Song HM, Woo J, Kim HK. In-vehicle network intrusion detection using deep convolutional neural network. Vehicular Communications. 2020;21:100198.

[86] Desta AK, Ohira S, Arai I, Fujikawa K. Rec-CNN: In-vehicle networks intrusion detection using convolutional neural networks trained on recurrence plots. Vehicular Communications. 2022;35:100470.

[87] Aliyu I, Feliciano MC, Van Engelenburg S, Kim DO, Lim CG. A Blockchain-Based Federated Forest for SDN-Enabled In-Vehicle Network Intrusion Detection System. IEEE Access. 2021;9:102593-608.

[88] Hacking, Lab CR. CAN Dataset for intrusion detection (OTIDS); 2020. Retrieved August 2021 from https://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset.

[89] Wang Y, Lai Y, Chen Y, Wei J, Zhang Z. Transfer learning-based self-learning intrusion detection system for in-vehicle networks. Neural Computing and Applications. 2023:1-17.

[90] Jedh M, Othmane LB, Ahmed N, Bhargava B. Detection of message injection attacks onto the can bus using similarities of successive messages-sequence graphs. IEEE Transactions on Information Forensics and Security. 2021;16:4133-46.

[91] Refat RUD, Elkhail AA, Hafeez A, Malik H. Detecting can bus intrusion by applying machine learning method to graph based features. In: Proceedings of SAI Intelligent Systems Conference. Springer; 2021. p. 730-48.

[92] Navet N, Song Y, Simonot-Lion F, Wilwert C. Trends in automotive communication systems. Proceedings of the IEEE. 2005;93(6):1204-23.

[93] Han ML, Kwak BI, Kim HK. Event-triggered interval-based anomaly detection and attack identification methods for an in-vehicle network. IEEE Transactions on Information Forensics and Security. 2021;16:2941-56.

[94] Hoang TN, Kim D. Detecting In-vehicle Intrusion via Semi-supervised Learning-based Convolutional Adversarial Autoencoders. arXiv preprint arXiv:220401193. 2022.

[95] Chockalingam V, Larson I, Lin D, Nofzinger S. Detecting attacks on the can protocol with machine learning. Annual EECS. 2016;588.

[96] Tomlinson A, Bryans J, Shaikh SA. Using a one-class compound classifier to detect in-vehicle network attacks. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion; 2018. p. 1926-9.

[97] Tomlinson A, Bryans J, Shaikh SA. Using internal context to detect automotive controller area network attacks. Computers & Electrical Engineering. 2021;91:107048.

[98] Taylor A, Leblanc S, Japkowicz N. Anomaly detection in automobile control network data with long short-term memory networks. In: 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA). IEEE; 2016. p. 130-9.

[99] Tanksale V. Anomaly detection for controller area networks using long short-term memory. IEEE Open Journal of Intelligent Transportation Systems. 2020;1:253-65.

[100] Novikova E, Le V, Yutin M, Weber M, Anderson C. Autoencoder anomaly detection on large CAN bus data. Proceedings of DLP-KDD. 2020.

[101] Kukkala VK, Thiruloga SV, Pasricha S. Indra: Intrusion detection using recurrent autoencoders in automotive embedded systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2020;39(11):3698-710.

[102] Longari S, Valcarcel DHN, Zago M, Carminati M, Zanero S. CANnolo: An anomaly detection system based on LSTM autoencoders for controller area network. IEEE Transactions on Network and Service Management. 2020;18(2):1913-24.

[103] Kukkala VK, Thiruloga SV, Pasricha S. LATTE: L STM Self-Att ention based Anomaly Detection in E mbedded Automotive Platforms. ACM Transactions on Embedded Computing Systems (TECS). 2021;20(5s):1-23.

[104] Thiruloga SV, Kukkala VK, Pasricha S. 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). IEEE; 2022. p. 326-31.

[105] Jeong S, Lee S, Lee H, Kim HK. X-CANIDS: Signal-Aware Explainable Intrusion Detection System for Controller Area Network-Based In-Vehicle Network. arXiv preprint arXiv:230312278. 2023.

[106] Balaji P, Ghaderi M. NeuroCAN: Contextual Anomaly Detection in Controller Area Networks. In: 2021 IEEE International Smart Cities Conference (ISC2). Manchester, United Kingdom: IEEE; 2021. p. 1-7.

[107] Narasimhan H, Vinayakumar R, Mohammad N. Unsupervised Deep Learning Approach for In-Vehicle Intrusion Detection System. IEEE Consumer Electronics Magazine. 2021.

[108] Wei P, Wang B, Dai X, Li L, He F. A novel intrusion detection model for the CAN bus packet of in-vehicle network based on attention mechanism and autoencoder. Digital Communications and Networks. 2023;9(1):14-21.

[109] He Y, Jia Z, Hu M, Cui C, Cheng Y, Yang Y. The hybrid similar neighborhood robust factorization machine model for can bus intrusion detection in the in-vehicle network. IEEE Transactions on Intelligent Transportation Systems. 2021.

[110] Tanaka D, Yamada M, Kashima H, Kishikawa T, Haga T, Sasaki T. In-vehicle network intrusion detection and explanation using density ratio estimation. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE; 2019. p. 2238-43.

[111] Sun H, Chen M, Weng J, Liu Z, Geng G. Anomaly detection for In-Vehicle network using CNN-LSTM with attention mechanism. IEEE Transactions on Vehicular Technology. 2021;70(10):10880-93.

[112] Zhang J, Li F, Zhang H, Li R, Li Y. Intrusion detection system using deep learning for in-vehicle security. Ad Hoc Networks. 2019;95:101974.

[113] Fenzl F, Rieke R, Chevalier Y, Dominik A, Kotenko I. Continuous fields: enhanced in-vehicle anomaly detection using machine learning models. Simulation Modelling Practice and Theory. 2020;105:102143.

[114] Martinelli F, Mercaldo F, Nardone V, Santone A. Car hacking identification through fuzzy logic algorithms. In: 2017 IEEE international conference on fuzzy systems (FUZZ-IEEE). IEEE; 2017. p. 1-7.

[115] Fenzl F, Rieke R, Dominik A. In-vehicle detection of targeted CAN bus attacks. In: The 16th International Conference on Availability, Reliability and Security. New York, NY, USA: Association for Computing Machinery; 2021. p. 1-7. Available from: https://doi.org/10.1145/3465481.3465755.

[116] Tomlinson A, Bryans J, Shaikh SA. Towards viable intrusion detection methods for the automotive controller area network. In: 2nd ACM Computer Science in Cars Symposium; 2018. p. 1-9.

[117] Longari S, Nova Valcarcel DH, Zago M, Carminati M, Zanero S. CANnolo: An Anomaly Detection System Based on LSTM Autoencoders for Controller Area Network. IEEE Transactions on Network and Service Management. 2021;18(2):1913-24.

[118] Wei P, Wang B, Dai X, Li L, He F. 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.

[119] Berger I, Rieke R, Kolomeets M, Chechulin A, Kotenko I. Comparative study of machine learning methods for in-vehicle intrusion detection. In: Computer Security. Springer; 2018. p. 85-101.

[120] Zhu K, Chen Z, Peng Y, Zhang L. Mobile edge assisted literal multi-dimensional anomaly detection of in-vehicle network using LSTM. IEEE Transactions on Vehicular Technology. 2019;68(5):4275-84.

[121] Gao L, Li F, Xu X, Liu Y. Intrusion detection system using SOEKS and deep learning for in-vehicle security. Cluster Computing. 2019;22(6):14721-9.

[122] Barletta VS, Caivano D, Nannavecchia A, Scalera M. Intrusion Detection for In-Vehicle Communication Networks: An Unsupervised Kohonen SOM Approach. Future Internet. 2020;12(7):119.

[123] Leslie N. An unsupervised learning approach for in-vehicle network intrusion detection. In: 2021 55th Annual Conference on Information Sciences and Systems (CISS). IEEE; 2021. p. 1-4.

[124] Lin Y, Chen C, Xiao F, Avatefipour O, Alsubhi K, Yunianta A. An evolutionary deep learning anomaly detection framework for in-vehicle networks-CAN bus. IEEE Transactions on Industry Applications. 2020.

[125] Nakamura S, Takeuchi K, Kashima H, Kishikawa T, Ushio T, Haga T, et al. In-Vehicle Network Attack Detection Across Vehicle Models: A Supervised-Unsupervised Hybrid Approach. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). IEEE; 2021. p. 1286-91.

[126] Hacking, Lab CR. Survival Analysis Dataset for automobile IDS; 2020. Retrieved August 2021 from https://ocslab.hksecurity.net/Datasets/survival-ids.

[127] Qin H, Yan M, Ji H. Application of Controller Area Network (CAN) bus anomaly detection based on time series prediction. Vehicular Communications. 2021;27:100291.

[128] Khan IA, Moustafa N, Pi D, Haider W, Li B, Jolfaei A. An enhanced multi-stage deep learning framework for detecting malicious activities from autonomous vehicles. IEEE Transactions on Intelligent Transportation Systems. 2021.

[129] Ashraf J, Bakhshi AD, Moustafa N, Khurshid H, Javed A, Beheshti A. Novel deep learning-enabled LSTM autoencoder architecture for discovering anomalous events from intelligent transportation systems. IEEE Transactions on Intelligent Transportation Systems. 2020;22(7):4507-18.

[130] Zhou W, Fu H, Kapoor S. CANGuard: Practical Intrusion Detection for In-Vehicle Network via Unsupervised Learning. In: 2021 IEEE/ACM Symposium on Edge Computing (SEC). IEEE; 2021. p. 454-8.

[131] Duan X, Yan H, Tian D, Zhou J, Su J, Hao W. In-Vehicle CAN Bus Tampering Attacks Detection for Connected and Autonomous Vehicles Using an Improved Isolation Forest Method. IEEE Transactions on Intelligent Transportation Systems. 2021.

[132] Dong C, Wu H, Li Q. Multiple Observation HMM-based CAN bus Intrusion Detection System for In-Vehicle Network. IEEE Access. 2023.

[133] Tian D, Li Y, Wang Y, Duan X, Wang C, Wang W, et al. An intrusion detection system based on machine learning for CAN-bus. In: International Conference on Industrial Networks and Intelligent Systems. Springer; 2017. p. 285-94.

[134] Wasicek A, Pesé MD, Weimerskirch A, Burakova Y, Singh K. Context-aware intrusion detection in automotive control systems. In: Proc. 5th ESCAR USA Conf; 2017. p. 21-2.

[135] Basavaraj D, Tayeb S. Towards a Lightweight Intrusion Detection Framework for In-Vehicle Networks. Journal of Sensor and Actuator Networks. 2022;11(1):6.

[136] Alshammari A, Zohdy MA, Debnath D, Corser G. Classification approach for intrusion detection in vehicle systems. Wireless Engineering and Technology. 2018;9(4):79-94.

[137] Khan MH, Javed AR, Iqbal Z, Asim M, Awad AI. DivaCAN: Detecting in-vehicle intrusion attacks on a controller area network using ensemble learning. Computers & Security. 2024:103712.

[138] Alfardus A, Rawat DB. 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). New York, NY, USA: IEEE; 2021. p. 0944-9.

[139] Amato F, Coppolino L, Mercaldo F, Moscato F, Nardone R, Santone A. CAN-bus attack detection with deep learning. IEEE Transactions on Intelligent Transportation Systems. 2021;22(8):5081-90.

[140] Dong Y, Chen K, Peng Y, Ma Z. Comparative Study on Supervised versus Semi-supervised Machine Learning for Anomaly Detection of In-vehicle CAN Network. arXiv preprint arXiv:220710286. 2022.

[141] Minawi O, Whelan J, Almehmadi A, El-Khatib K. 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. DIVANet '20. New York, NY, USA: Association for Computing Machinery; 2020. p. 41–47. Available from: https://doi.org/10.1145/3416014.3424581.

[142] Anjum A, Agbaje P, Hounsinou S, Olufowobi H. In-Vehicle Network Anomaly Detection Using Extreme Gradient Boosting Machine. In: 2022 11th Mediterranean Conference on Embedded Computing (MECO). Budva, Montenegro: IEEE; 2022. p. 1-6.

[143] Park S, Choi JY. Hierarchical anomaly detection model for in-vehicle networks using machine learning algorithms. Sensors. 2020;20(14):3934.

[144] Kalkan SC, Sahingoz OK. 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). IEEE; 2020. p. 1-6.

[145] Moulahi T, Zidi S, Alabdulatif A, Atiquzzaman M. Comparative Performance Evaluation of Intrusion Detection Based on Machine Learning in In-Vehicle Controller Area Network Bus. IEEE Access. 2021.

[146] Zhang L, Shi L, Kaja N, Ma D. A two-stage deep learning approach for can intrusion detection. In: Proc. Ground Vehicle Syst. Eng. Technol. Symp.(GVSETS); 2018. p. 1-11.

[147] Zhang L, Ma D. A hybrid approach toward efficient and accurate intrusion detection for in-vehicle networks. IEEE Access. 2022;10:10852-66.

[148] Weber M, Klug S, Sax E, Zimmer B. Embedded hybrid anomaly detection for automotive CAN communication. In: 9th European Congress on Embedded Real Time Software and Systems (ERTS 2018); 2018. .

[149] Müter M, Groll A, Freiling FC. A structured approach to anomaly detection for in-vehicle networks. In: 2010 Sixth International Conference on Information Assurance and Security. IEEE; 2010. p. 92-8.

[150] Pevnỳ T. Loda: Lightweight on-line detector of anomalies. Machine Learning. 2016;102(2):275-304.

[151] Kang DM, Yoon SH, Shin DK, Yoon Y, Kim HM, Jang SH. A Study on Attack Pattern Generation and Hybrid MR-IDS for In-Vehicle Network. In: 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC). IEEE; 2021. p. 291-4.

[152] Suda H, Natsui M, Hanyu T. Systematic intrusion detection technique for an in-vehicle network based on time-series feature extraction. In: 2018 IEEE 48th International Symposium on Multiple-Valued Logic (ISMVL). IEEE; 2018. p. 56-61.

[153] Khan Z, Chowdhury M, Islam M, Huang Cy, Rahman M. Long Short-Term Memory Neural Network-based Attack Detection Model for In-Vehicle Network Security. IEEE Sensors Letters. 2020.

[154] Kaiser Christian FA Stocker Alexander. Automotive CAN bus data: An Example Dataset from the AEGIS Big Data Project; 2020. Retrieved August 2021 from https://zenodo.org/record/3267184#.YRB6m1NKijR.

[155] Xiao J, Wu H, Li X. Internet of things meets vehicles: sheltering in-vehicle network through lightweight machine learning. Symmetry. 2019;11(11):1388.

[156] Ma H, Cao J, Mi B, Huang D, Liu Y, Li S. A GRU-Based Lightweight System for CAN Intrusion Detection in Real Time. Security and Communication Networks. 2022;2022.

[157] NasrEldin A, Bahaa-Eldin AM, Sobh MA. In-Vehicle Intrusion Detection Based on Deep Learning Attention Technique. In: 2021 16th International Conference on Computer Engineering and Systems (ICCES). IEEE; 2021. p. 1-7.

[158] Hossain MD, Inoue H, Ochiai H, Fall D, Kadobayashi Y. LSTM-Based Intrusion Detection System for In-Vehicle Can Bus Communications. IEEE Access. 2020;8:185489-502.

[159] Hossain MD, Inoue H, Ochiai H, Fall D, Kadobayashi Y. An effective in-vehicle CAN bus intrusion detection system using CNN deep learning approach. In: GLOBECOM 2020-2020 IEEE Global Communications Conference. IEEE; 2020. p. 1-6.

[160] Tariq S, Lee S, Woo SS. CANTransfer: transfer learning based intrusion detection on a controller area network using convolutional LSTM network. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing; 2020. p. 1048-55.

[161] Mehedi ST, Anwar A, Rahman Z, Ahmed K. Deep transfer learning based intrusion detection system for electric vehicular networks. Sensors. 2021;21(14):4736.

[162] Taslimasa H, Dadkhah S, Neto ECP, Xiong P, Iqbal S, Ray S, et al. ImageFed: Practical Privacy Preserving Intrusion Detection System for In-Vehicle CAN Bus Protocol. In: 2023 IEEE 9th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). IEEE; 2023. p. 122-9.

[163] Kishore C, Rao DC, Behera H, et al. 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. Springer; 2022. p. 531-44.

[164] Rehman A, Rehman SU, Khan M, Alazab M, Reddy T. CANintelliIDS: Detecting In-Vehicle Intrusion Attacks on a Controller Area Network using CNN and Attention-based GRU. IEEE Transactions on Network Science and Engineering. 2021.

[165] Lo NW, Tsai HC. Illusion attack on vanet applications-a message plausibility problem. In: 2007 IEEE Globecom Workshops. IEEE; 2007. p. 1-8.

[166] Loukas G, Vuong T, Heartfield R, Sakellari G, Yoon Y, Gan D. Cloud-based cyber-physical intrusion detection for vehicles using deep learning. Ieee Access. 2017;6:3491-508.

[167] Cho KT, Shin KG. Viden: Attacker identification on in-vehicle networks. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: Association for Computing Machinery; 2017. p. 1109-23. Available from: https://doi.org/10.1145/31339.

[168] Choi W, Joo K, Jo HJ, Park MC, Lee DH. Voltageids: Low-level communication characteristics for automotive intrusion detection system. IEEE Transactions on Information Forensics and Security. 2018;13(8):2114-29.

[169] Xun Y, Zhao Y, Liu J. VehicleEIDS: A Novel External Intrusion Detection System Based on Vehicle Voltage Signals. IEEE Internet of Things Journal. 2021.

[170] Oseni A, Moustafa N, Janicke H, Liu P, Tari Z, Vasilakos A. Security and privacy for artificial intelligence: Opportunities and challenges. arXiv preprint arXiv:210204661. 2021.

[171] Foruhandeh M, Man Y, Gerdes R, Li M, Chantem T. SIMPLE: Single-frame based physical layer identification for intrusion detection and prevention on in-vehicle networks. In: Proceedings of the 35th annual computer security applications conference; 2019. p. 229-44.

[172] Li Y, Lin J, Xiong K. An Adversarial Attack Defending System for Securing In-Vehicle Networks. In: 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC); 2021. p. 1-6.

[173] Khan Z, Chowdhury M, Islam M, Huang CY, Rahman M. In-vehicle false information attack detection and mitigation framework using machine learning and software defined networking. arXiv preprint arXiv:190610203. 2019.

[174] Lin HC, Wang P, Chao KM, Lin WH, Chen JH. Using Deep Learning Networks to Identify Cyber Attacks on Intrusion Detection for In-Vehicle Networks. Electronics. 2022;11(14):2180.

[175] Han ML, Kwak BI, Kim HK. Anomaly intrusion detection method for vehicular networks based on survival analysis. Vehicular communications. 2018;14:52-63.

[176] Hacking, Lab CR. Car Hacking Attack and Defense Challenge; 2020. Retrieved August 2021 from https://ocslab.hksecurity.net/Datasets/carchallenge2020.

[177] Hacking, Lab CR. CAN Signal Extraction and Translation Dataset; 2020. Retrieved July 2022 from https://ocslab.hksecurity.net/Datasets/can-signal-extraction-and-translation-dataset.

[178] Hanselmann M, Strauss T, Dormann K, Ulmer H. SynCAN Dataset; 2020. Retrieved August 2021 from https://github.com/etas/SynCAN/blob/master/README.md.

[179] TU Eindhoven DoM, Science C. TU Eindhoven CAN bus intrusion dataset; 2020. Retrieved August 2021 from https://doi.org/10.4121/uuid:b74b4928-c377-4585-9432-2004dfa20a5d.

[180] of Cryptography L, Security S. CrySyS Lab dataset; 2020. Retrieved August 2021 from https://www.crysys.hu/research/vehicle-security/.

[181] Luo F, Wang J, Zhang X, Jiang Y, Li Z, Luo C. In-vehicle network intrusion detection systems: a systematic survey of deep learning-based approaches. PeerJ Computer Science. 2023;9:e1648.

[182] Verma M, Bridges R, Hollifield S. ACTT: Automotive CAN tokenization and translation. In: 2018 International Conference on Computational Science and Computational Intelligence (CSCI). IEEE; 2018. p. 278-83.

[183] Torrey L, Shavlik J. Transfer learning. In: Handbook of research on machine learning applications and trends: algorithms, methods, and techniques. IGI global; 2010. p. 242-64.

[184] Vinyals O, Blundell C, Lillicrap T, Wierstra D, et al. Matching networks for one shot learning. Advances in neural information processing systems. 2016;29:3630-8.

[185] Xian Y, Schiele B, Akata Z. Zero-shot learning-the good, the bad and the ugly. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017. p. 4582-91.

[186] Li Y, Lin J, Xiong K. An adversarial attack defending system for securing in-vehicle networks. In: 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC). IEEE; 2021. p. 1-6.

[187] Zhang X, Zhu X, Lessard L. Online data poisoning attacks. In: Learning for Dynamics and Control. PMLR; 2020. p. 201-10.

[188] Blevins DH, Moriano P, Bridges RA, Verma ME, Iannacone MD, Hollifield SC. Time-based can intrusion detection benchmark. arXiv preprint arXiv:210105781. 2021.

[189] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781. 2013.

[190] Liu X, Zhang F, Hou Z, Mian L, Wang Z, Zhang J, et al. Self-supervised learning: Generative or contrastive. IEEE transactions on knowledge and data engineering. 2021;35(1):857-76.

[191] Rajapaksha S, Kalutarage H, Al-Kadri MO, Petrovski A, Madzudzo G. Improving in-vehicle networks intrusion detection using on-device transfer learning. In: Proceedings of the Symposium on Vehicles Security and Privacy, San Diego, CA, USA. vol. 27; 2023. .

[192] Adewole KS, Salau-Ibrahim TT, Imoize AL, Oladipo ID, AbdulRaheem M, Awotunde JB, et al. Empirical analysis of data streaming and batch learning models for network intrusion detection. Electronics. 2022;11(19):3109.

[193] Viegas E, Santin A, Abreu V, Oliveira LS. Stream learning and anomaly-based intrusion detection in the adversarial settings. In: 2017 IEEE Symposium on Computers and Communications (ISCC). IEEE; 2017. p. 773-8.

[194] Nixon C, Sedky M, Hassan M. Reviews in Online Data Stream and Active Learning for Cyber Intrusion Detection-A Systematic Literature Review. In: 2021 Sixth International Conference on Fog and Mobile Edge Computing (FMEC). IEEE; 2021. p. 1-6.

[195] Li H, Zhao L, Juliato M, Ahmed S, Sastry MR, Yang LL. Poster: Intrusion detection system for in-vehicle networks using sensor correlation and integration. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security; 2017. p. 2531-3.

[196] Sharmin S, Mansor H, Abdul Kadir AF, Aziz NA. Using Streaming Data Algorithm for Intrusion Detection on the Vehicular Controller Area Network. In: Inernational Conference on Ubiquitous Security. Springer; 2021. p. 131-44.

[197] Papernot N, McDaniel P, Swami A, Harang R. Crafting adversarial input sequences for recurrent neural networks. In: MILCOM 2016 - 2016 IEEE Military Communications Conference; 2016. p. 49-54.

[198] James N, Ong LY, Leow MC. Exploring Distributed Deep Learning Inference Using Raspberry Pi Spark Cluster. Future Internet. 2022;14(8):220.

[199] Mirzadeh SI, Farajtabar M, Pascanu R, Ghasemzadeh H. Understanding the role of training regimes in continual learning. Advances in Neural Information Processing Systems. 2020;33:7308-20.

[200] Desta AK, Ohira S, Arai I, Fujikawa K. ID Sequence Analysis for Intrusion Detection in the CAN bus using Long Short Term Memory Networks. In: 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops); 2020. p. 1-6.

[201] Droździel P, Tarkowski S, Rybicka I, Wrona R. Drivers 'reaction time research in the conditions in the real traffic. Open Engineering. 2020;10(1):35-47. Available from: https://doi.org/10.1515/eng-2020-0004 [cited 2022-11-28].

[202] Tolpegin V, Truex S, Gursoy ME, Liu L. Data poisoning attacks against federated learning systems. In: Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25. Springer; 2020. p. 480-501.

[203] Vinutha H, Poornima B, Sagar B. Detection of outliers using interquartile range technique from intrusion dataset. In: Information and Decision Sciences: Proceedings of the 6th International Conference on FICTA. Springer; 2018. p. 511-8.

[204] Kromanis R, Kripakaran P. Support vector regression for anomaly detection from measurement histories. Advanced Engineering Informatics. 2013;27(4):486-95.

[205] Cinà AE, Grosse K, Demontis A, Biggio B, Roli F, Pelillo M. Machine Learning Security against Data Poisoning: Are We There Yet? arXiv preprint arXiv:220405986. 2022.

[206] Tomlinson A, Bryans J, Shaikh SA, Kalutarage HK. Detection of automotive CAN cyber-attacks by identifying packet timing anomalies in time windows. In: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W). IEEE; 2018. p. 231-8.

[207] Stabili D, Marchetti M, Colajanni M. Detecting attacks to internal vehicle networks through Hamming distance. In: 2017 AEIT International Annual Conference. IEEE; 2017. p. 1-6.

[208] Verma ME, Iannacone MD, Bridges RA, Hollifield SC, Moriano P, Kay B, et al. Addressing the lack of comparability & testing in CAN intrusion detection research: A comprehensive guide to CAN IDS data & introduction of the ROAD dataset. arXiv preprint arXiv:201214600. 2020.

[209] Lokman SF, Othman AT, Musa S, Abu Bakar MH. Deep contractive autoencoder-based anomaly detection for in-vehicle controller area network (CAN). In: Progress in Engineering Technology. Springer; 2019. p. 195-205.

[210] Park H, Noh J, Ham B. Learning memory-guided normality for anomaly detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2020. p. 14372-81.

[211] Angiulli F, Fassetti F, Ferragina L. LatentOut: an unsupervised deep anomaly detection approach exploiting latent space distribution. Machine Learning. 2022:1-27.

[212] ElMorshedy MM, Fathalla R, El-Sonbaty Y. Feature Transformation Framework for Enhancing Compactness and Separability of Data Points in Feature Space for Small Datasets. Applied Sciences. 2022;12(3):1713.

[213] Bergsma W. A bias-correction for Cramér's V and Tschuprow's T. Journal of the Korean Statistical Society. 2013;42(3):323-8. Available from: https://www.sciencedirect.com/science/article/pii/S1226319212001032.

[214] Moriano P, Bridges RA, Iannacone MD. Detecting CAN Masquerade Attacks with Signal Clustering Similarity. arXiv preprint arXiv:220102665. 2022.

[215] Ganesan A, Rao J, Shin K. Exploiting consistency among heterogeneous sensors for vehicle anomaly detection. SAE Technical Paper; 2017.

[216] Acock AC, Stavig GR. A measure of association for nonparametric statistics. Social Forces. 1979;57(4):1381-6.

[217] Bergsma W. A bias-correction for Cramér's V and Tschuprow's T. Journal of the Korean Statistical Society. 2013;42(3):323-8.

[218] Akoglu H. User's guide to correlation coefficients. Turkish journal of emergency medicine. 2018;18(3):91-3.

[219] Marchetti M, Stabili D. READ: Reverse engineering of automotive data frames. IEEE Transactions on Information Forensics and Security. 2018;14(4):1083-97.

[220] Ladjal S, Newson A, Pham CH. A PCA-like autoencoder. arXiv preprint arXiv:190401277. 2019.

Appendix A

CAN-MIRGU dataset

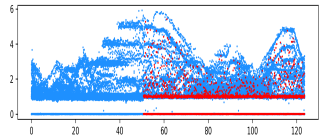
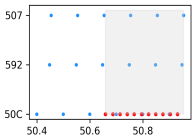
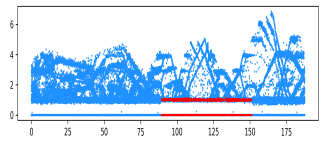
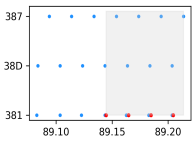
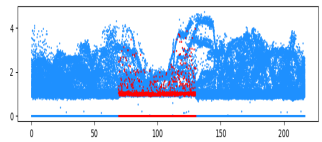
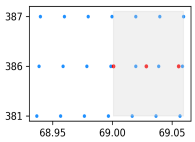
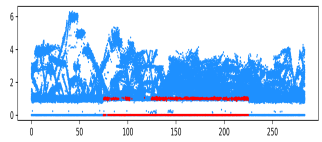
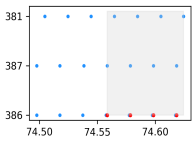
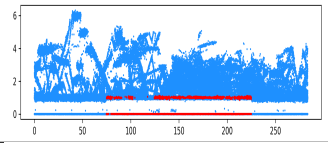
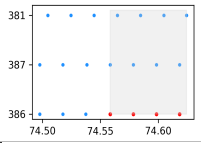
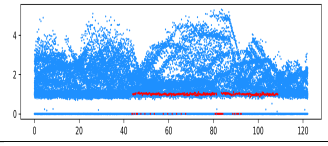
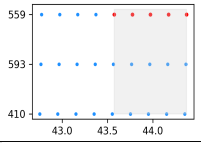
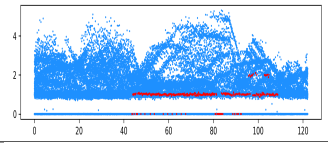
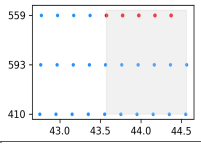
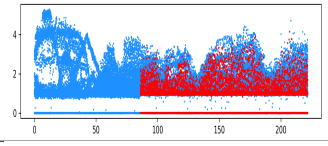
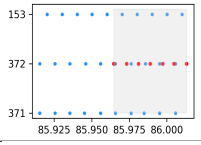
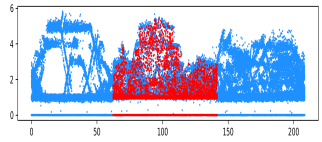
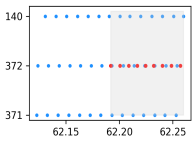
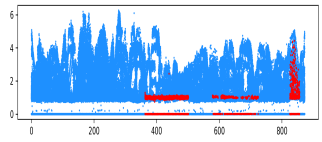
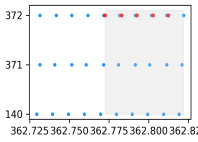
Attack	Observations	Message Timing	Targeted ID Message Timing
Drive mode changing ★ 50C#FF05FFFF24FFFFE0 123.605818 Injecting every 0.02s	Continuously switching between normal, sport, eco and eco+ driving modes for a few seconds and stabled at eco+.		
Power steering ★ 381#FFB73FXXXXXXXXXX 187.484292 Flam	'Check motor-driven power steering' warning message on the dashboard, slightly less control of the steering wheel.		
Max speedometer ★ 386#FFFFFFFFFFFFFFFF 216.432840 Injecting every 0.02s	Speedometer jumps to 159 mph while driving at 30 mph		
Min speedometer 1 ★ 386#FF027002F9821D42 283.422522 Flam	Speedometer jumps to 15 mph while driving at 30 mph		
Min speedometer masquerade ★ 386#FF027002F9821D42 283.422522 Flam	Simulated attack. It is expected to have a comparable impact to the Min speedometer 1 attack.		
Wiper warning ★ 559#XXXXXCXXXXXXXXXX 122.107031 Flam	Set the front wiper speed to 2 on the dashboard. No physical movement of the wiper.		
Wiper warning masquerade ★ 559#XXXXXCXXXXXXXXXX 122.107031 Masquerade	Simulated attack. It is expected to have a comparable impact to the wiper warning attack.		
Gear shifter attack 1 ★ 372#8000010000000AA05 221.688076 Injecting every 0.001s	'Shifting not possible due to overheating' warning message, Steering wheel becomes stiffer.		
Gear shifter attack 2 ★ 372#0000010000000AA05 208.340552 Injecting every 0.001s	'Shifting not possible due to overheating' warning message. Steering wheel became too loose.		
Multiple attacks 1 ★ 372#XXFFXXXXXXXXXXXX 559#XXXXXCXXXXXXXXXX 386#00000000F982FFFF 872.579076 Flam and Injecting every 0.02s	Changed driving mode into 2WD certification mode for ID 372 attack, set the front wiper speed to 2 on the dashboard for ID 559 attack, speedometer jumps to 19 mph while driving at 30 mph.		

Table A.1: Description of attacks

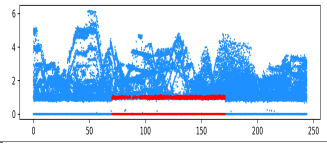
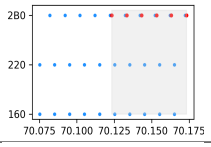
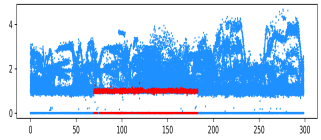
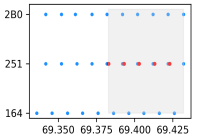
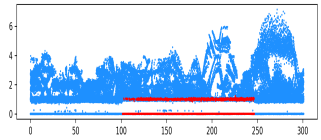
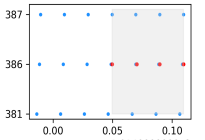
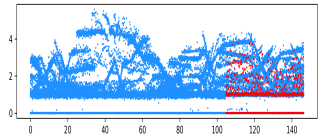
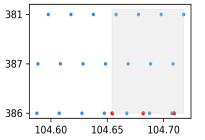
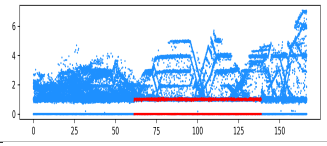
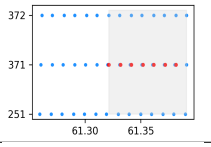
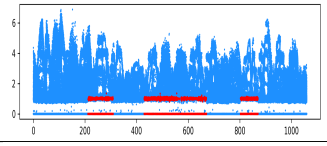
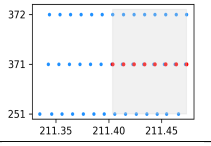
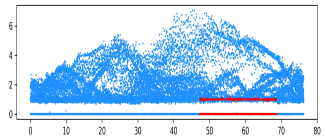
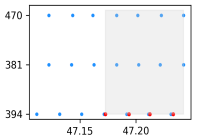
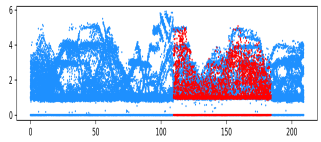
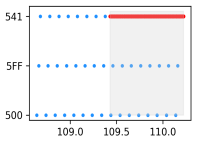
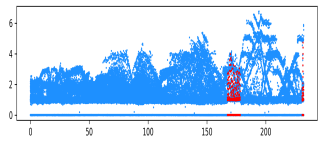
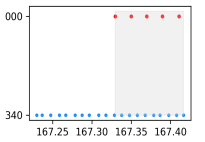
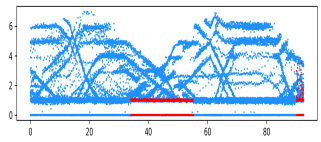
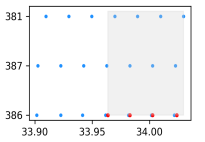
Attack	Observations	Message Timing	Targeted ID Message Timing
Steering angle replay ★ 2B0#0700000755 243.840814 Flam	No visible changes		
FCA warning attack ★ 251#008DXXXXXXXXXXXX 298.643430 Flam	'Check brake light' and 'Check fog light' warning messages on the dashboard and continuous warning sounds		
Min speedometer 2 ★ 386#0100B4821783FCC2 300.926023 Flam	Speedometer jumps to 13 mph while driving at 30 mph		
Min speedometer 3 ★ 386#FF027002F9821D42 146.289314 Injecting every 0.02s	Speedometer varies between 21-27 mph while driving at 30 mph		
EMS ★ 371#371FXXXXXX2BXXXX 166.409611 Flam	No visible changes		
EMS replay long ★ 371#2E1E000000000010 1058.974900 Flam	No visible changes		
Parking brake ★ 394#XXXXXXXXXXXXAAXX 76.243699 Flam	Warning sounds		
Door open warning ★ 541#03114100000A0045 209.072355 Injecting every 0.001s	Front and rear door open warning message on the dashboard and warning sounds		
Fuzzing valid IDs and DoS ★ XXX#FFFFFFFFFFFFFFFF 232.154914 Injecting every 0.02s	No changes during DoS, Warning messages and sounds including driving mode changes during valid IDs fuzzing		
Reverse speedometer and fuzzing ★ 386#0100B4821783FCC2 XXX#FFFFFFFF00FFFFFF 92.542629 Flam and Injecting every 0.02s	Speedometer jumps to 13 mph while driving at 5 mph for speedometer attack, Change driving mode into sport for fuzzing attack		

Table A.2: Description of attacks

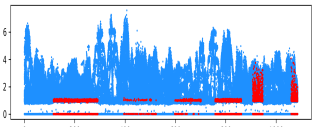
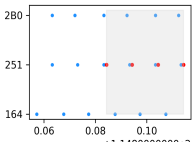
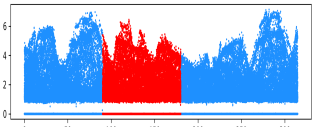
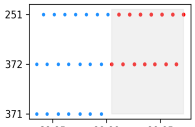
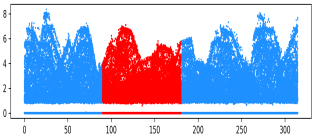
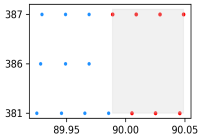
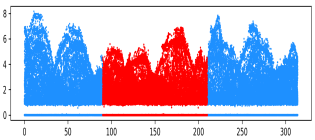
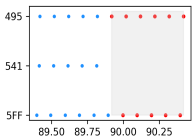
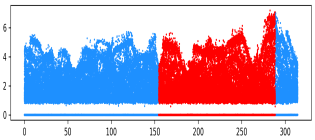
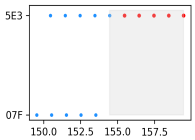
Attack	Observations	Message Timing	Targeted ID Message Timing
Multiple attacks 2 ★ 251#FFFFXXXXXXXXXXXX 07F#FFFFXXXXXXXXXXXX 593#01XXXXXXXXXX 160#02AAXXXXXXXXXXXX XXX#FFFFFFFFFFFFFFFF XXX#FFFFFFFF00FFFFFF 1085.432364 Flam and Injecting every 0.02s	'Check FCA(Forward Coll Avoidance Assist)' warning message and sound for ID 0x251, 'Check break light' and 'Check fog light' warning messages and sounds for ID 0x07F, 'Check tyre pressure monitoring system' warning message and sound for ID 0x593, 'Stop vehicle and check breaks' warning message and sound for ID 0x160, Different warning messages and sounds including drive mode change for Fuzzing random IDs and Fuzzing valid IDs attacks		
ID 371 suspension ★ 371#XXXXXXXXXXXXXXXX 314.692548 Suspension	Simulated attack.		
ID 386 suspension ★ 386#XXXXXXXXXXXXXXXX 314.516615 Suspension	Simulated attack.		
ID 541 suspension ★ 541#XXXXXXXXXXXXXXXX 313.716066 Suspension	Simulated attack.		
ID 07F suspension ★ 07F#XXXXXXXXXXXXXXXX 314.360475 Suspension	Simulated attack.		

Table A.3: Description of attacks