Intrusion Detection for In-Vehicle CAN Communication Using Deep Neural Networks

FRIDA SUNDFELDT AND BIANCA WIDSTAM
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Intrusion Detection for In-Vehicle CAN Communication Using Deep Neural Networks

Frida Sundfeldt and Bianca Widstam

Department of Electrical and Information Technology
Lund University

Supervisor: Christian Gehrmann and Mahshid Helali Moghadam

Examiner: Thomas Johansson

May 27, 2024

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

As the automotive industry progresses with the development of connected vehicles, this digital evolution comes with inherent cyber security risks and thus securing vehicle communication systems, particularly the Controller Area Network (CAN) bus, is crucial. International regulations such as UN Regulation No 155, necessitate cyber security measures for threat detection, prevention, and mitigation. In this regard, vehicle manufacturers are mandated to implement measures to detect and prevent cyber-attacks against vehicles. Thus, onboard Intrusion Detection Systems (IDSs) for in-vehicle networks, e.g. the CAN bus, can help detect various cyber-attacks with different mechanisms such as Fabrication (e.g., Denial of Service and Fuzzy), Suspension, Masquerade (e.g. Spoofing attack), and Replay. Network IDSs provide a layer of security by monitoring and analyzing the data traffic, and identifying suspicious activities that could indicate an intrusion. To address this, the thesis, first, introduces a tool that generates attack data by analyzing normal data files, including both open-source and proprietary data provided by Scania—collected from a test vehicle. Along with that, it uses real-world labeled open-source attack datasets to identify realistic patterns that correspond to various types of attacks. Following this analysis, the process involves altering the proprietary dataset to mimic real attack scenarios closely. This tool will aid with testing the IDS's effectiveness in detecting attacks on the CAN bus. Secondly, the thesis investigates the viability of a machine learning-based IDS, using two different supervised deep learning models, Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM), to identify attacks on the CAN bus. An empirical evaluation of the models is performed—considering sequence inputs with various lengths—and the results indicate that with a longer sequential input, more instances of Spoofing and Replay attacks, as more complex attack classes, can be correctly detected. Moreover, despite showing comparable accuracy, the LSTM models can lead to a slightly higher rate of misclassified normal states as attacks compared to CNN. Furthermore, the validity of the synthetically generated data, the limitations, and the importance of developing models that can adapt to new, unknown attack types are also elaborated and discussed.

Populärvetenskaplig sammanfattning

Så CAN maskininlärning upptäcka attacker mot fordon

Med ökande uppkoppling i fordon växer risken för cyberattacker, vilket hotar både individens och allmänhetens säkerhet. Vårt examensarbete förbättrar säkerheten inom fordon genom att utveckla och testa ett maskininlärningsbaserat intrångsdetekteringssystem för CAN-bussen, med syntetiskt genererad och verklig fordonsdata från Scania.

Våra bilar blir allt smartare, men även mer sårbara för cyberattacker. Moderna fordon är uppbyggda av många elektroniska styrenheter (ECUs), som hanterar alla möjliga typer av funktioner. Dessa enheter kommunicerar med varandra genom det så kallade CAN-bussystemet, en teknik som liknar internet där meddelanden utbyts mellan varje styrenhet. Till skillnad från dagens internet, som har avancerade säkerhetsåtgärder som kryptering och autentisering, saknar CAN många av dessa skydd. Detta öppnar upp för en rad potentiella hot, som exempelvis skadliga meddelandeinjektioner som kan påverka bilens kritiska funktioner. Forskare har bevisat att det är möjligt att manipulera bland annat bilens broms och hastighetsfunktioner genom att ta kontroll över någon av styrenheterna eller bilens diagnosuttag (OBD-II) och skicka felaktiga meddelanden.

För att bemöta dessa hot har vi utvecklat ett maskininlärningsbaserat intrångsdetekteringssystem (IDS) som använder två typer av djupinlärningsmodeller, CNN (Convolutional Neural Network) och LSTM (Long Short-Term Memory). En IDS kan effektivt identifiera anomala mönster i CAN-trafiken som kan innebära att en meddelandeinjektionsattack har skett.

Genom att analysera CAN-data insamlad från ett testfordon hos Scania och jämföra denna med öppen källdata från ytterligare tre fordon, har vi skapat flera filer med syntetisk attackdata. Denna data simulerar CAN-kommunikation under olika typer av cyberattacker och har använts för att träna våra modeller. Dessa lär sig att känna igen tecken på att systemet är under attack, vilket gör det möjligt för dem att detektera och klassificera olika cyberhot med en noggrannhet på upp till 99,84 %.

Acknowledgement

We would like to extend special thanks to our supervisor at Scania, Mahshid Helali Moghadam. Thank you for the engaging discussions and for encouraging and challenging us throughout the process.

We would also like to thank our supervisor at LTH, Christian Gehrmann, for your valuable input and guidance. Additionally, we are grateful to Thomas Johansson for agreeing to be our examiner.

Lastly, we want to thank Scania and the wonderful people we had the opportunity to work with. Thank you for hosting us and providing the tools and support we needed.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

As the automotive industry progresses with advancements such as autonomous and connected vehicles, the complexity and functionality of in-vehicle systems have significantly increased. Vehicles today are not only modes of transport but also vast networks of computers forming advanced electronic ecosystems. Modern vehicles can be equipped with up to 100 of these computers, known as Electronic Control Units (ECUs), which are interconnected through various communication networks such as the Controller Area Network (CAN), LIN (Local Interconnect Network), and Ethernet. These systems support a broad range of functionalities, from critical ones such as braking to less critical ones like media systems [2]. While the transformation from purely mechanical vehicles to highly connected ones offers many benefits in terms of safety and features, it also expands the vehicle's attack surface, making it susceptible to a variety of cyber threats. The digitalization involves not only the functionalities of the vehicle but also connects the vehicle with its surroundings, for example, through WiFi or Bluetooth. This new development has been implemented with a focus on vehicle safety rather than security [3]. The CAN bus, which is the standard network protocol within most vehicles, lacks many of the security measures employed on the internet, such as encryption and authentication [4]. This makes it susceptible to, for example, message injection attacks. Over the past decade, several successful attacks have been conducted by researchers on the CAN bus, through the OBD-II diagnostics port on the vehicle, and through WiFi and Bluetooth. These attacks have enabled researchers to control critical functionalities such as the brakes and the steering wheel [3].

These alarming discoveries have forced the automotive industry to prioritize the development of robust cybersecurity frameworks. This is reflected in regulatory standards such as the UN Regulation No. 155 [5], which mandates comprehensive cybersecurity measures to detect and counteract threats effectively. In response to these security vulnerabilities, the automotive industry is turning toward more advanced methods to improve the security of the CAN protocol. While many approaches have focused on physical security measures like restricting access and employing simple encryption, these methods are often insufficient due to the lightweight and basic nature of the ECUs and CAN bus [6].

To tackle this issue more effectively, a solution could be to install Machine Learning (ML)-based Intrusion Detection Systems (IDSs) in vehicles. An IDS can analyze network traffic patterns and detect unusual activities that could indicate a cyberattack. By using machine learning, the IDS can efficiently identify a wide range of threats [6]. However, one significant challenge with implementing an IDS in vehicles is that it needs to learn from CAN data to effectively recognize and respond to cyber threats. The scarcity of available CAN traffic data, especially datasets that include real attack scenarios, poses a substantial hurdle. This lack of data limits the ability of an IDS to be effectively trained and tested [7].

1.1 Goal

The goal of this thesis is to investigate cyber threats to in-vehicle CAN buses and explore how these threats can be detected using an ML-based IDS. The study will concentrate on a set of common attacks categorized under the mechanisms of Masquerade, Suspension, Fabrication, and Replay. By analyzing the patterns for each attack mechanism along with real CAN data captured from a test truck in a normal state, the first part of the thesis involves the creation of a set of synthetic attack files. To mimic the data distribution of attack messages in real scenarios, open source files of real attacks will be analyzed. Secondly, the IDS will be implemented by exploring two different deep learning models, which will be trained and tested with the synthetically created data. For clarification, in this thesis, real CAN data is defined as data captured directly from a vehicle. A real attack on a CAN bus includes actual interference, such as modifying a node to send malicious frames or removing a node that normally transmits. Conversely, a synthetic attack involves altering CAN logs after collection.

The goal of the thesis is divided into the following two research questions.

1.2 Research Questions

RQ.1 How to generate CAN traffic data with a set of common cyber attacks e.g., Masquerade, Suspension, Fabrication, and Replay, based on a log file containing normal traffic data?

RQ.2 How can deep learning anomaly detection techniques be used to develop an effective IDS?

1.3 Outline

The project has been carried out in two parts which also reflects the outline of the report. The initial chapter introduces the topic, outlines the thesis objectives, and poses the research questions. The second chapter provides a comprehensive background on the CAN bus, IDS, and relevant attack methods. Following this, the report moves into the first part of the thesis, focusing on the creation and

implementation of a CAN attack data generator. This section details the process of synthetically generating attack data. The fourth chapter shifts to the second part of the thesis, which involves developing and implementing an ML-based IDS by exploring two different deep learning models and discussing the methods used for implementing, training, and evaluating these models. The fifth chapter showcases the outcomes from both sections, with a discussion of these results in the sixth chapter. The thesis concludes with a final chapter seven.

Chapter 2
Background

2.1 Controller Area Network (CAN)

The Controller Area Network (CAN) bus is the standard communication network for reliable, real-time message delivery within a vehicle. Initially intended for vehicles, the CAN bus standard has been further integrated into various control systems. It was originally developed by Robert Bosch GmbH in 1983, published the CAN 2.0 specification A and B in 1991 [8], and in 1993, the protocol was adopted as an ISO standard (ISO 11898) [9]. The standard CAN protocol facilitates interaction between the vehicle's sensors and Electronic Control Units (ECUs) and supports a data rate of up to 1 Mbps with a maximum payload size of 8 bytes. CAN Flexible Data-rate (CAN-FD) was introduced in 2012 [10], and was standardized in ISO 11898-1:2015 [11], allowing for higher data rates and larger payload sizes. CAN-FD enables data transmission rates between ECUs beyond 1 Mbps and increasing the maximum payload size from 8 bytes to 64 bytes. A modern vehicle typically consists of about 70 ECUs, each responsible for controlling various functions of the vehicle [12]. These include the Engine Control Module, Airbag Control Module, and Anti-Lock Brake System, among others.

2.1.1 OSI layers

The message-based protocol standard consists of multiple abstraction layers, with the most significant being the physical and data link layer of the OSI model.

Physical layer

The physical layer defines the hardware required for a CAN network and holds for example cables and electrical signal levels. The CAN protocol enables a one-point-of-entry communication between ECUs that avoids complex wiring. The CAN network operates as a dual-wire serial bus, connecting each ECU to these two wires. Data transmission occurs through the CAN High (CAN_H) and CAN Low (CAN_L) lines, with the dominant bit denoting a logic "zero" and the recessive bit a logic "one" [13]. In the recessive state, both lines maintain a voltage of approximately 2.5V. Conversely, in the dominant state, CAN_H rises to a higher voltage level, and CAN_L drops to a lower voltage level. This method allows ECUs to change the CAN bus to a dominant state for signal transmission, while

in the absence of signals, terminating resistors automatically revert the bus to a recessive state.

Data link layer

The data link layer transmits these messages over the CAN bus via CAN frames and is responsible for example, message framing, arbitration, error detection, and acknowledgment. A CAN message can transmit up to eight bytes of data and the length is specified in the data length code (DLC). Two CAN frame formats are commonly used, the standard/base format (CAN 2.0A) and the extended frame format (CAN 2.0B), which differ in the bit length of the identifier [8]. The standard format supports an 11-bit unique identifier while the extended frame supports a 29-bit, consisting of the 11-bit identifier (the base identifier) and an 18-bit extension (the identifier extension). The CAN-FD frame supports both the 11-bit identifier and the extended, but the payload size has been increased to up to 64 bytes. Vehicle manufacturers maintain the detailed semantics of the CAN ID and data field as proprietary and confidential. The CAN protocol allows for message prioritization based on identifiers, with lower identifiers receiving higher priority. The composition of the standard CAN and CAN-FD frame, as shown in Figure 2.1 and Figure 2.2 includes the following key components:

- **Arbitration Field**: Includes the ID identifier and RRS bit (Remote Transmission Request), determining the message priority, with lower ID values indicating higher priority. The RRS bit determines if it is a data or remote frame. Data frames transmit data, while remote frames are used to request data.
- **Control Field:** Manages various data frame aspects such as Data Length Code (DLC) indicating the size of the payload.
- **Data Field**: In standard CAN, this field can carry up to 8 bytes of data, translating to a limited number of signals. CAN-FD extends this to 64 bytes, indicating a larger number of signals within a single message.
- **Cyclic Redundancy Check (CRC) Field**: Used for transmission error detection, it contains a CRC value that the receiving ECU checks against its own calculation to confirm message accuracy.
- **Acknowledgment (ACK) Field**: Receivers use this to acknowledge that the data frame was received.

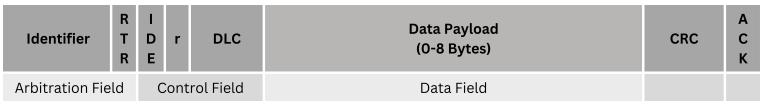


Figure 2.1: The Standard CAN (2.0) Data Frame

Figure 2.2: The CAN Flexible Data-rate (CAN-FD) Data Frame

2.1.2 Vulnerabilities

The CAN standard has multiple advantages, e.g., built-in error detection, cost-effectiveness due to less wiring, and lightweight design, but it also has security vulnerabilities. These weaknesses include a lack of mechanisms for authentication, authorization, and encryption [13]. The current design assumes that every message that is sent to an ECU is legitimate. Without authentication and authorization, it's impossible to verify the origin of a message, allowing malicious nodes to impersonate legitimate ECUs using their arbitration IDs. Implementing authentication, while necessary for security, demands more advanced hardware, which not only increases costs but may also introduce latency problems [13]. Furthermore, the lack of encryption means that all CAN traffic is transparent and allows attackers to easily intercept and analyze the data. Since messages on the CAN bus are broadcast, a single compromised node could grant an attacker access to all the information transmitted on the bus.

2.2 Attack Models

The lack of security mechanisms in the CAN bus makes it susceptible to a range of attacks. This was proven, among others, by Kosher et al. [3] who performed physical access and non-physical access attacks on the CAN bus where they could successfully circumvent a wide range of safety-critical systems and control various automotive functions. There are three representative categories of attack mechanisms identified by Cho and Shin, Fabrication, Suspension, and Masquerade, that can severely compromise the functionality of the in-vehicle system [14]. Therefore, at least one attack from each category has been considered in this thesis. Additionally, a Replay attack has been included to increase variation and robustness as it presents challenges in detection. Together, these attacks provide a broad spectrum of threats, ensuring that the IDS is tested against both common and severe cyber threats, thus enhancing its efficacy and reliability in real-world scenarios. The following sections provide an introduction to the attack models in the literature that are relevant to this thesis.

2.2.1 Fabrication Attacks

In a Fabrication attack, an attacker, through control of an in-vehicle ECU, fabricates and injects malicious messages with falsified ID, DLC, and data. This type of attack typically targets overriding messages sent by a legitimate safety-critical ECU, leading to potential malfunctions or inoperative states in recipient ECUs. Using high-priority messages to dominate the CAN network, as seen in DoS attacks, exemplifies one attack within the spectrum of Fabrication attacks. In Figure 2.3a, one simple Fabrication attack is displayed where the compromised ECU C introduces multiple attack messages with ID A with a high frequency, usually sent by ECU A. As a result, ECUs expecting messages with ID A receive these malicious messages more frequently than legitimate ones.

Denial of Service (DoS)

In a Denial of Service (DoS) attack in vehicles, lots of requests overwhelm the system, potentially disrupting its functionality and causing other ECUs to stop or delay their transmission. The attacker injects high-priority CAN messages, e.g. using the 0x000 CAN ID, in a high frequency on the CAN bus [6, 15]. Flooding the CAN bus with these high-priority messages denies legitimate messages from being transmitted since ID 0x000 is guaranteed to win arbitration and is rarely used by legitimate ECUS.

Fuzzy Attack

The Fuzzy attack involves injecting messages with random or semi-random CAN IDs and data values. Through this method, an attacker can insert malicious data into the network, using randomly faked identifications. Furthermore, by systematically testing with small fuzzy packets and observing the behavior of the CAN bus, the attacker can gather critical information for creating targeted attacks [3]. This technique eliminates the need for the attacker to have prior knowledge of reverse engineering or the specific components of the vehicle.

2.2.2 Suspension Attack

In a Suspension attack, the attacker targets a compromised ECU to prevent it from transmitting some or all messages, that could cause malfunctions in the compromised ECU and other dependent ECUs. For example, if the electric power steering ECU stops sharing steering angle data, the electronic stability control system, needing that information for traction control, fails to operate correctly [14]. Figure 2.3b illustrates the Suspension attack mechanism where ECU C has been compromised and suspended from transmitting messages.

2.2.3 Masquerade Attack

In a Masquerade attack, two ECUs are compromised: a weaker one, which can be stopped or suspended from sending messages but cannot inject fabricated messages, and a stronger one, fully controlled by the attacker for injecting

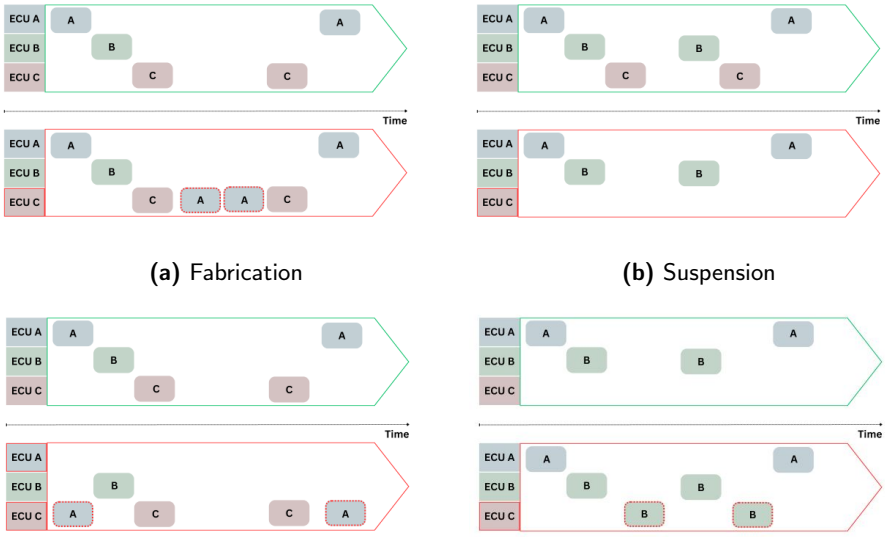
malicious messages. The attacker mimics the message pattern of the target (weaker) ECU, stops its transmissions, and starts sending forged messages using the stronger ECU. This type of attack can cause significant disruptions, without changing the frequency of message transmissions, posing a critical threat to vehicular safety and network integrity. Figure 2.3c shows an example of the Masquerade attack mechanism where ECU A (weaker) and ECU B (stronger) have been compromised. ECU A has been suspended from transmitting any messages and ECU B mimics ECU A by transmitting malicious messages with ID A.

Spoofing Attack

In a Spoofing attack on the CAN bus, the attacker injects malicious messages that appear to be from another ECU on the network. The attacker creates and transmits fake messages with a spoofed ID, which are then interpreted as authentic by other ECUs within the system. The messages are created to either mimic the normal communication from a legitimate ECU or to send malicious commands or data. In this thesis, the Spoofing attack includes the Suspension of the compromised ECU, classifying it as a Masquerade attack mechanism. This approach not only introduces malicious activity but also stop the activity from the legitimate ECU, increasing the attack's disruption.

2.2.4 Replay Attack

The Replay attack is achieved by capturing real-time CAN messages and then replaying the message. In this attack, the intruder first listens and records data packets during normal behavior of the network, here the intruder can analyze and look into the meaning of the messages and target those that trigger specific functions on the vehicle. These recorded messages are then replayed at a later time. In this way, the intruder can cause malfunction to the vehicle without the need for deep knowledge of the system. Since the Replay attack uses unaltered messages these attacks can be harder to detect. Figure 2.3d shows ECU B's messages being captured and replayed by ECU C, disrupting the intended sequence of operations.



(a) Fabrication **(b)** Suspension

(c) Masquerade **(d)** Replay

Figure 2.3: Figures illustrating four examples of CAN bus attack mechanisms. Each shows normal message flow (the upper green arrow) and during attacks (the lower red arrow). Compromised ECUs and injected attack messages are highlighted with a red border.

2.3 Intrusion Detection System (IDS)

Intrusion Detection Systems (IDSs) are used to identify unauthorized access or attacks on industrial networks or computer systems [16, 17]. They function by monitoring the network traffic and system activities, and by analyzing these parameters they can detect malicious patterns or anomalies that suggest an attack has happened.

In the context of the security vulnerabilities of the CAN bus system, IDSs have been identified as a leading defensive mechanism against malicious attacks [18, 19]. The theory of using an IDS in the automotive system is a widely researched topic and Lokman et al. have proposed a taxonomy for the categories of deployment strategies and detection approaches [20]. Deployment strategies refer to the placement and integration of IDS within the vehicle, i.e., how and where these systems should be implemented. Meanwhile, detection approaches describe the various techniques used by the IDS to identify potential threats.

2.3.1 Deployment Strategies

Lokman et al. present three possible deployment strategies for the IDS in a vehicle: the CAN bus, the ECUs, and central gateways [20]. These strategies are classified into two categories: host-based IDS and network-based IDS. Host-based IDS means installing the IDS directly on an ECU, and since the ECUs broadcast messages on the CAN bus this allows the IDS to monitor all traffic. Network-based IDS refers to deploying the IDS either on the CAN bus itself or at a central gateway, positioning it within the network where it can also monitor all traffic. Each of these deployment strategies faces limitations, for instance, deployment on an ECU is constrained by the ECU's available computational power and memory, limiting the complexity of the implemented IDS algorithm. Furthermore, deploying an IDS on an individual ECU limits the monitored traffic to that specific ECU's CAN bus. In contrast, placing the IDS on the CAN bus itself or at a central gateway allows it to monitor traffic across multiple CAN buses, including traffic passing through gateways.

2.3.2 Detection approaches

The functionality of the IDS depends on its detection approach. The different detection approaches proposed by Lokman et al. [20] are under the categories of signature-, anomaly-, and specification-based approaches.

Signature-based approach

The signature-based approach focuses on defined patterns, or signatures, within analyzed data. By comparing network activity against predefined attack patterns stored in the IDS, this method predicts well-known attacks very effectively. However, it is not capable of detecting new, unknown intrusions and must frequently update its signature database with new attacks. Song et al. [21] introduced a signature-based approach based on the analysis of time intervals of CAN messages that detect injection attacks without false positive errors.

Anomaly-based approach

Another approach is to analyze the message traffic of the CAN bus and look for irregular patterns or behavior. This approach has been widely investigated in the literature such as in [22, 1, 23, 24]. Anomaly-based detection techniques observe network activity and compare it against recorded normal behavior. If the deviation reaches a specified threshold, an attack is detected. After an extensive training phase, this method can effectively detect new attacks but is prone to generating false positives on normal packets, as it considers anything that deviates from normal behavior as an intrusion. To establish a model of normal behavior, earlier researchers have typically employed frequency-based, machine learning-based, and statistical-based techniques.

Frequency-based method is based on monitoring deviations from known frequencies and timing of CAN messages. It relies on the predictability of the

CAN bus, where CAN messages are broadcast at fixed time intervals and frequency [20]. The method looks for the regularity of packet intervals and any attack that mimics this behavior will bypass detection. Additionally, irregular or unpredictable packet frequencies, often caused by noise in the CAN bus, may result in increased false negatives.

Machine learning-based method refers to when an IDS leverages machine learning techniques. This method utilizes models trained on specific CAN bus features, such as message traffic. These models are usually categorized into supervised, unsupervised, or even semi-supervised types [20]. Each of these types offers different strengths in detecting and responding to potential threats. Unsupervised and semi-supervised models are considered anomaly-based detection, while supervised learning can also be considered signature-based detection because it relies on predefined labels that can represent known attack signatures.

Supervised models involve a training process where the model learns from a labeled dataset that includes both normal and anomaly data. Multiple supervised approaches have been proposed in the literature such as Bari et al. [6], that compared three supervised models including Support Vector Machine (SVM), Decision Tree (DT), and K-Nearest Neighbor (KNN) to detect and classify intrusions on the CAN bus. Their approach uses labeled input and output data. This approach is good at detecting the specific types of attacks for which it has been trained, which gives it a high accuracy for known threats. However, since it is only trained on a set of anomaly patterns it has a limitation in identifying attacks outside of its training data, making it ineffective against unknown attacks or zero-day attacks [25]. Further, to achieve a good result, the model requires a large dataset of labeled data to be constructed, which can be time-consuming [20, 22].

In contrast, unsupervised models are trained on data representing the normal behavior of the system, without any anomalies. This method enables the model to learn a baseline of how the system acts in a normal state and can compare data with anomalies to spot deviations. The advantage of this approach is its ability to detect a wide variety of attacks, including new and unknown ones [26]. However, the implementation is more complex and difficult due to the need to establish an optimal threshold for distinguishing between normal state data and anomaly data. Unsupervised models perform worse because they produce a high false-positive rate [22].

A semi-supervised model integrates the properties of both supervised and unsupervised approaches. It combines the high accuracy detection capabilities of supervised learning for known threats with the new and unknown anomaly detection capabilities of unsupervised learning. This gives the hybrid model a high accuracy when identifying familiar patterns with the ability to detect previously unseen threats without needing a large amount of labeled data [22].

Statistical-based method uses statistical properties to monitor and identify deviations from normal patterns in the CAN network. This could involve assessing parameters such as mean, variance, and standard deviation to detect attacks. For example, [14] involves analyzing the physical characteristics of the bus, using variations among ECUs to detect unauthorized modifications or access. The IDS uses Recursive Least Squares (RLS) to model the expected clock skews of the ECUs and uses Cumulative Sum (CUSUM) analysis to detect any significant deviations from these models.

Hybrid-based method integrates multiple detection methods that can collectively monitor and detect various aspects of the CAN bus, for example combining specification-based and machine learning-based. This method can detect a wide range of intrusions but poses challenges such as model training time and redundancy [20].

Specification-based approach

The specification-based method uses explicit specifications that describe normal behaviors in terms of system components such as the CAN protocol or ECUs. It detects attacks when there is a deviation from these designated specifications. Unlike anomaly-based approaches, which for example use statistical models to define normal behavior, the specifications in this approach are often manually defined by human experts. Larson et al. [27] gather information from the CANopen standard protocol and object directory sections to detect cyber attacks within the in-vehicle network. They demonstrated that potential attacks could be identified by analyzing extracted information from specifications, and also concluded that gateway ECUs are the most likely targets for attackers. Olufowobi et al. [28] present another specification-based approach that investigates specific expectations for the timing behaviors on the CAN bus network. Their paper effectively detects data injection attacks with low false positive rates.

2.3.3 Deep learning

Deep learning (DL) is increasingly used for anomaly detection in-vehicle security. DL is a part of machine learning that involves the use of artificial neural networks (ANNs) with multiple layers, known as deep neural networks [13]. ANNs are inspired by the structural and functional aspects of biological brains and aim to simulate these processes in a computational model, using a layered architecture of neurons. The architecture is particularly effective due to its depth, which refers to the number of hidden layers between the input and output layers. CAN IDS can use deep learning's ability to autonomously extract data features and adapt to real-time parameters, improving their accuracy in anomaly detection. Given the varying characteristics of input data, deep learning offers different types of architectures, such as convolutional neural networks and recurrent neural networks.

Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is one type of deep learning architecture that is specifically designed to process two-dimensional data, such as images [29]. The basic CNN architecture is made of several layers such as convolutional layers, pooling layers, and fully-connected layers. The convolution layers use filters to generate feature maps highlighting spatial features in the input. The pooling layers reduce feature map dimensions by executing downsampling while preserving essential features. There are several types of pooling layers, e.g., max pooling, min pooling, and average pooling. For example, max pooling picks the maximum value from a narrow window of the feature map and ignores the other values, making it less sensitive to small changes in the input and spatial translations. Finally, the fully-connected layers perform linear and non-linear transformations to produce the final network output. Additionally, CNNs may have other layers such as dropout and batch normalization layers to enhance convergence and generalization. The architecture design, including the choice of the number of layers, filter sizes, and activation functions has significantly impacted the model's complexity and generalization abilities. The activation function introduces non-linearity and decides which neurons that should be activated, i.e., determine how much information to pass to the next layer. There are multiple types of activation functions used in CNN systems, such as sigmoid and Rectified Linear Unit (ReLU). The mathematical formulas for these can be seen in Equation 2.1 and 2.2. For more detail on CNN architectures, see [29].

$$f(x) = \frac{1}{1 + \exp(-x)} \tag{2.1}$$

$$f(x) = \max(0, x) \tag{2.2}$$

Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM)

Recurrent Neural Network (RNN) is a deep learning model that uses sequential data, such as words, sentences, and audio. RNNs use cyclic connections with recurrent layers or hidden layers of recurrent cells, which allows updating their current state based on both past states and current input data [30]. This sequential memory is important for tasks where context from earlier in the sequence is necessary to understand or predict later elements. RNNs usually consist of standard recurrent cells (i.e., sigma and tanh cells). The equations of a simple recurrent sigma cell are given by Equations 2.3 and 2.4

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b) \tag{2.3}$$

$$y_t = h_t \tag{2.4}$$

where:

- h_t represents the recurrent information at time t
- σ is the activation function (in this example, the sigmoid function)

- W_h and W_x are the weight matrices for the hidden state and input state
- x_t is the input at time t
- b is the bias
- y_t, is the final output of the network at time t, from the current state h_t

Despite their advantages, RNNs often encounter difficulties when learning long-term dependencies due to issues such as error signals that propagate backward in time and tend to either exponentially increase or vanish. Long Short-Term Memory (LSTM) is a type of RNN that addresses this issue and can handle the long-term dependencies in sequential data. It was introduced by Hochreiter and Schmidhuber in 1997 [31]. LSTMs accomplish long-term dependencies by introducing gate functions into the cell structure, including the input gate, forget gate, and output gate, which regulate the flow of information through the network. It is important to note that there are variants of the LSTM architecture. The first version did not include a forget gate, it was introduced by Gers et al. in 2000 [32]. The input gate controls what new information is stored in the cell state, the forget gate discards information that is no longer relevant, and the output gate determines what information is output based on the cell state [30]. The LSTM cell with a forget gate is given by Equation 2.5 and is based on Figure 2.4, which is adapted of [30].

$$\begin{aligned} f_t &= \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f) \\ i_t &= \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i) \\ \tilde{c}_t &= \tanh(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c})} \\ c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \\ o_t &= \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o) \\ h_t &= o_t \cdot \tanh(c_t) \end{aligned} \tag{2.5}$$

where:

- h_{t-1} is the hidden state from the previous timestep
- x_t is the input at the current timestep
- $W_{fh}, W_{ih}, W_{\tilde{c}h}, W_{oh}$ are the weight matrices for the hidden state from the previous timestep
- $W_{fx}, W_{ix}, W_{\tilde{c}x}, W_{ox}$ are the weight matrices for the input at the current timestep
- $b_f, b_i, b_{\tilde{c}}, b_o$ are the biases for each gate
- σ is the sigmoid activation function
- tanh is the hyperbolic tangent function, used to normalize cell updates and output transformations
- $\cdot$ is the pointwise multiplication of two vectors

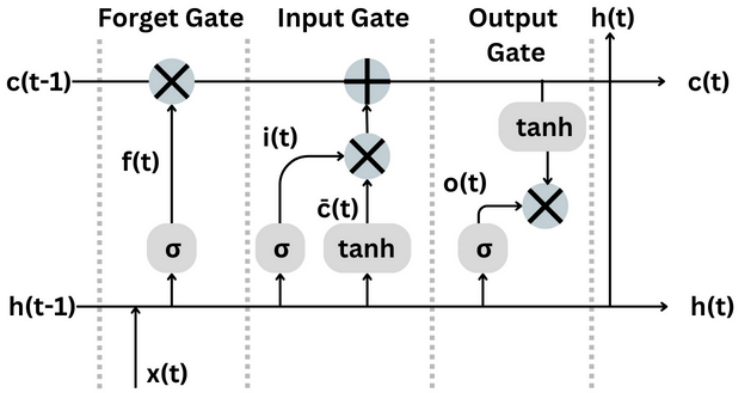


Figure 2.4: LSTM cell with forget gate

2.4 Previous Research

This section introduces previous research related to this thesis, including existing datasets with recorded CAN data, attack generation tools, and deep learning IDS. The section finishes with a contribution statement.

2.4.1 Existing datasets

For effective IDS implementation, it is important to have access to valid and realistic data. However, the availability of publicly accessible datasets containing real CAN data of high quality, particularly those including verified and labeled attacks, is limited. The lack of real attack data for moving vehicles is due to high expenses, time-consuming processes, and potential safety risks during its generation.

Verma et al. [15] present a Real ORNL Automotive Dynamometer (ROAD) CAN IDS dataset, consisting of over 3.5 hours of one vehicle's CAN data including various attacks. These include real attacks such as Fuzzy, Fabrication, unique advanced attacks, and simulated Masquerade attacks. It further provides a comprehensive guide of existing CAN IDS datasets, categorizes CAN attacks, and assesses the datasets' quality and suitability for research. One of the disadvantages of this dataset is that attack intervals are labeled rather than each message, making it difficult to use when creating attack data for this thesis.

The datasets containing labeled attack data come with their own limitations. The Car Hacking Dataset for Intrusion Detection released by HCRL [33] from the research in [23], includes normal behavior vehicle data as well as vehicle data including three types of attacks, namely DoS, Fuzzy, and Spoofing. This dataset was used to train and build a deep-learning IDS. The normal vehicle data included around 1 million messages and was obtained during 8 minutes. The

attack data included around 4 million messages and was collected between 45 to 90 minutes. The reason why this dataset is unsuitable to use to test an IDS is that the normal behavior data was obtained during a driving session, whereas the attack data was gathered while the vehicle was stationary, making them substantially different. Additionally, there is a large time gap, with the longest being 22 seconds, in the attack data where no messages appear to be transmitted, indicating the need for dataset pruning before being used. This observation was made during the evaluation of the data for this thesis and also noticed by Verma et al. [15] in their dataset evaluation. The distinct characteristics of the normal behavior and attack data, coupled with the significant time gap, led to the decision not to use this dataset for creating the synthetic attack datasets for this thesis.

The dataset analyzed and used to create the attack datasets for this thesis is the HCRL Survival Analysis Dataset for Automobile IDS, presented by Han et al. [1]. This dataset includes Flooding (DoS), Fuzzy, and Malfunction attacks. In this dataset, normal and attack driving data were obtained from three different types of vehicles and were used for an anomaly IDS based on the survival analysis model. Although the dataset's duration is relatively short, ranging only from 60 to 90 seconds, it provides real attack scenarios on multiple vehicles, repetition of the same attacks across these vehicles, and the attack data is labeled per message, which makes it valuable for the purpose of this thesis.

2.4.2 Attack generation

Many versions of attack generation tools are available, designed to generate datasets with injected attacks. Huang et al. [7] introduce an Attack Traffic Generation (ATG) tool that simulates four main types of attacks: DoS, Fuzzy, Spoofing, and attacks exploiting CAN bus error handling mechanisms. This attack data is intended for evaluating security mechanisms developed for CAN systems. It offers dataset generation with support for flexible attack configuration, as well as reading and replaying of CAN messages. It provides an open source graphical Python application that uses hardware such as the primary supported USB2CAN device, one of the cheapest options on the market. The application is presented as a cheaper option than established CAN tools like CANoe and CarShark. This ATG tool pre-configures a set of attacks and is injected into the log files at set intervals. In contrast, the attack generator tool in this thesis synthetically injects varying attacks into CAN logs based on an analysis of the given log file and using a DBC file, without the need for hardware.

The Attack Traffic Generation software, described in the thesis by Neelap and Bhandari [34], is designed to generate synthetic attack traffic for testing Network-based IDS within CAN environments. This ATG analyzes CAN logs to generate attack traffic similar to this thesis. The implemented attacks include Fuzzy, Spoofing, Replay, and Overwrite. It highlights the vulnerabilities of CAN systems to cyberattacks and the significance of analyzing message patterns, frequencies, and intervals within CAN system traffic to understand log file

dynamics for realistic injections of attack. The thesis by Neelap and Bhandari demonstrates the software's ability to inject malicious traffic into normal data using pattern analysis and randomness, mimicking real-life cyberattacks. To generate data for the IDS in this thesis, a similar approach of extracting information from log files and using this analysis to generate attack data is used. This functionality is further extended by analyzing open source files and obtaining information about the sending ECUs by decoding the messages with a DBC file and with the addition of a Suspension attack.

2.4.3 Deep Learning IDS

Hossain et al. [35] propose a supervised LSTM-based IDS to detect cyberattacks on in-vehicle CAN bus systems. The IDS uses the raw payload, CAN ID, and DLC of CAN messages as input features for the model. To train and test their model, the authors generated a dataset from an experimental car, injecting DoS, Fuzzy, and Spoofing attacks. They employed the LSTM model for supervised binary and multiclass classification, achieving a high accuracy of 99.995%. They selected an LSTM because it performs well with time-series data and sequence classification. Additionally, they compared their LSTM with the Survival Analysis dataset [1], which is used in this master thesis to create realistic attack scenarios. The paper explores different hyper-parameter values, and the best-performing parameters have been used to develop the LSTM-based IDS described in this thesis. However, the methods differ as the LSTM in this thesis processes sequences of messages, while their LSTM model analyzes sequences of features from a single message.

In a subsequent study, Hossain et al. [36] employed a supervised CNN for a similar purpose but expanded the testing to three different car models. Like the LSTM, this CNN-based IDS analyzes the CAN bus data features, payload, CAN ID, and DLC. Their experiment demonstrated a detection accuracy of 99.99% and a detection rate (recall) of 0.99. This paper concludes that deep-learning-based intrusion detection systems are more effective than other methods. The model achieved a 100% detection accuracy for DoS and Spoofing attacks while Fuzzy attacks had a slightly lower accuracy. They concluded that the challenge in detecting Fuzzy attacks stems from the presence of thousands of random CAN IDs. The CNN model was chosen due to its effectiveness in processing time-series data and extracting features from raw inputs, thus being able to detect subtle attack signatures that other methods might fail to notice. Furthermore, the study showed the importance of filter size in the performance of the IDS, demonstrating that 256 and 512 filter sizes were more effective than smaller sizes due to their balance of accuracy and low variance. Based on these findings, this thesis has adopted a 256-filter setting for the 1D CNN model to optimize detection capability.

Hoang and Kim [22] introduce an approach to an in-vehicle IDS using semi-supervised learning through Convolutional Adversarial Autoencoders (CAA). The proposed model is designed to detect various message injection

attacks, including DoS, Spoofing, and Fuzzy similar to the investigation into common cyber threats in this thesis. Their model requires only a small amount of labeled data—about 10% of the training dataset which reduces the time spent on data collection. They used CAN IDs as features because the CAN IDs sequence follows a pattern, and injected messages break this pattern. Therefore, their model can capture normal and abnormal patterns for correct classification. The CAN IDs are represented in a 29-bit format, as in this thesis, to enhance efficiency and ensure the model adapts to any version of CAN messages. The input is generated by stacking 29 consecutive CAN IDs, each represented in a 29-bit format, to form a 29 x 29 matrix. Their model achieved a F1 score of 0.9984 and a low error rate of 0.1% with limited labeled data.

2.4.4 Contributions

The main contributions of this thesis are summarized as follows:

- Development of an attack generator that synthetically injects various attacks into CAN logs. The injection pattern for the Fabrication attack scenario is derived from the analysis of open source real attack datasets. Moreover, augmenting the attack generation process by using a CAN Database (DBC) file, which contains information for decoding raw CAN data, enables more realistic generation of the Suspension and Masquerade attack scenarios.
- Training and testing of two deep learning models using real test truck data derived from a driving session that includes CAN-FD data, along with synthetically injected attacks.
- Comparison of two deep learning models, CNN and LSTM, in detecting attacks against the CAN bus.
- Analysis of how different sequence lengths of CAN bus data impact the performance of CNN and LSTM models in detecting injected attacks.

Chapter 3
Attack Generator

This chapter outlines the development of an extended attack generator based on earlier research [34] including the attack models discussed in section 2.2. The attack generator implemented in this thesis further takes into account patterns from open source log files to inject messages in a realistic way (i.e., introducing a more variable attack pattern using normal distribution for timing). Developed in Python, the attack generator tool extracts parameters and patterns, while also mapping message IDs to the sending Electric Control Unit (ECU) from real vehicle data. Based on this analysis and attack patterns derived from existing literature the tool injects messages simulating an attack and outputs a new log file. The resulting log files, including a wide range of attack scenarios, will be used in training and assessing the Machine Learning (ML)-based Intrusion Detection System (IDS). Further, the chapter concludes with an evaluation methodology to assess the performance of the generator itself. The effectiveness of the generated attack data is measured by how similar it is to already existing attack datasets in terms of the ratio of attack messages in the intrusions.

3.1 Design overview

The attack generator, composed of Python scripts, is designed to process normal behavior in-vehicle Controller Area Network (CAN) data, from both open source datasets and a proprietary dataset provided by Scania. The initial script converts different log file formats into a standardized Comma-Separated Values (CSV) format, ensuring uniform column headers (timestamp, arbitration ID, DLC, data, type). The attack generator script then parses the CSV files containing the CAN messages into a DataFrame using the Python library Pandas. The Pandas library offers several advantages for handling and analyzing complex and large datasets which is necessary due to the size of the datasets used in this thesis project. An analysis is carried out using the normal data datasets, both open source and from Scania, together with labeled open source attack data to identify realistic patterns that correspond to various types of attacks. Following this analysis, the process involves altering the Scania dataset to closely mimic a real attack scenario on that specific dataset. This tweaking is done by modifying the normal CAN traffic data contained in a CSV file through the injection of new messages, alteration of existing ones, or the removal of

messages, depending on the specific characteristics of the intended attack.

For every simulated attack, the system generates a corresponding CSV file. These files contain synthetically produced data that based on the analysis show how the proprietary data would look under that attack. This manipulation provides a way of representing an attack without actually simulating it on a real CAN bus. A visual representation of the design is depicted in Figure 3.1.

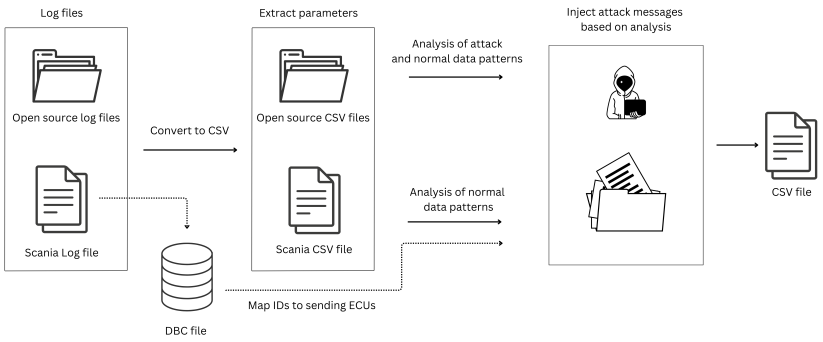


Figure 3.1: Attack Generator Design

In the following sections of the chapter, the specifics of each attack generated by the framework will be discussed. These parts will detail the methods and mechanisms used to alter the Scania dataset, thereby synthetically simulating these attacks.

3.2 Dataset description

3.2.1 Scania Dataset

The dataset used in simulating synthetic attacks is normal behavior CAN bus data provided by Scania in the format of an ASCII log file, as well as a CAN DBC (CAN database) file. The DBC file is used to decode raw CAN bus data into meaningful physical values, enabling the identification of transmitting nodes within the network. This proprietary data, collected from a real driving session in a Scania truck, spans approximately 19 minutes and contains around 12 million messages. The presence of CAN-FD messages in the log shows that the vehicle's CAN bus system is compatible with CAN-FD, supporting larger data payloads up to 64 bytes and higher data rates compared to the standard CAN protocol. The data attributes for each message extracted from the log file include a timestamp, CAN ID, Data Length Code (DLC), and data bytes. Each attribute is briefly explained as follows:

- Timestamp: The recorded time in seconds

- CAN ID: The arbitration ID used to identify the CAN message in hexadecimal format (29-bits)
- DLC: The number of data bytes ranging from 0 to 64
- Data: The payload of the message (byte) in hexadecimal format

This traffic data is valuable due to the limitation of datasets derived from real driving sessions, offering realistic payload behavior as sensors capture actual values (e.g., pedestrian crossings, and lane assistance metrics).

3.2.2 Survival Analysis Dataset (KIA, SONATA, SPARK)

The Survival Analysis Dataset for automobile IDS [1], includes real driving data (normal and attack) that were collected through an in-vehicle OBD-II port from three types of vehicles. The data is focused on the following attack scenarios: overloading the network (DoS) and injecting random packets (Fuzzy). The attack data was generated by injected attack packets for five seconds every 20 seconds for each of the three scenarios. The amount of messages in every dataset can be seen in Table 3.1. The log files only include standard CAN messages with a payload of up to 8 bytes. The data attributes include a timestamp, CAN ID, DLC, data bytes, and a flag indicating whether a message is normal (R) or injected (T). The brief explanations of the attributes are as follows:

- Timestamp: The recorded time in seconds
- CAN ID: The arbitration ID used to identify the CAN message in hexadecimal format (11-bit, e.g., 018F)
- DLC: The number of data bytes ranging from 0 to 8
- Data: The payload of the message (byte) in hexadecimal format
- Flag: T or R, T defines an attack message while R represents a normal message

Table 3.1: Number of Messages by Attack Type and Vehicle Model [1]

Attack Type	**# of Msg (SONATA)**	**# of Msg (KIA)**	**# of Msg (SPARK)**
Normal data	117,173	192,516	136,934
DoS Attack	149,547	181,901	120,570
Fuzzy Attack	135,670	249,990	65,665

3.3 CAN Traffic Analysis

The initial phase in the development of the attack generator involved processing CAN message data. This involved converting the raw ASCII log file (.asc) into a more accessible CSV format (.csv). Only the essential attributes were preserved – i.e., the timestamp, arbitration identifier, and data field. All CAN messages in the log file were decoded using DBC files, and stored in a dictionary format where each ECU was mapped to its corresponding sending message IDs.

As in the attack traffic generator developed by Neelap and Bhandari [34], message ID frequencies and timestamps were targeted and analyzed from the proprietary data. Additionally, analyzing the interval between messages in open source data, when there is an attack, can reveal patterns that can be applied to the generator. The proprietary data and the open source data differ significantly in aspects such as vehicle type, diversity of driver activities, and duration. Furthermore, the open source data uses a standard CAN protocol while the proprietary data uses CAN-FD. These variations can, for example, result in differences in message frequency on the CAN bus. To accurately compare attack data from both datasets, it's important to normalize this data relative to the normal traffic patterns. To achieve this, a scale factor is used to ensure that comparisons are meaningful and consider the inherent differences between the datasets.

Table 3.2: Average interval between every message in the normal datasets

Dataset	Proprietary	SONATA	KIA	SPARK
Avg. interval (ms)	0.0952	0.5121	0.4785	0.4351

Table 3.2 shows that the average interval between all messages is smaller in the proprietary data than in the open source data. The scale factor is calculated from the average intervals obtained using the expression in 3.1.

$$\begin{aligned} \text{Scale Factor} &= \frac{A_{\text{PROP.}}}{\frac{1}{3}\left(A_{\text{KIA}} + A_{\text{SONATA}} + A_{\text{SPARK}}\right)} \\ &= \frac{0.0952}{\frac{1}{3}\left(0.5121 + 0.4785 + 0.4351\right)} \\ &\approx 0.2 \end{aligned} \tag{3.1}$$

This scale factor is used to find what the mean interval should be for the synthetically created DoS and Fuzzy attack, i.e., how often messages should be injected into the normal data. The interval in which the messages should be injected is calculated by applying the scale factor to the average interval between attack messages in the open source data. The resulting intervals can be seen in Table 3.3.

Table 3.3: Average interval between attack messages in Fuzzy and DoS

	Scaled interval	SONATA	KIA	SPARK
DoS	0.1260	0.6183	0.6059	0.6655
Fuzzy	0.1948	1.0142	0.9746	0.9334

For the Replay attack the most frequent arbitration IDs are analyzed and chosen as target messages for the attack. Figure 3.2 shows the frequency of messages per ID in decreasing order, five IDs are sending messages with the highest frequency and these are targeted for the attack.

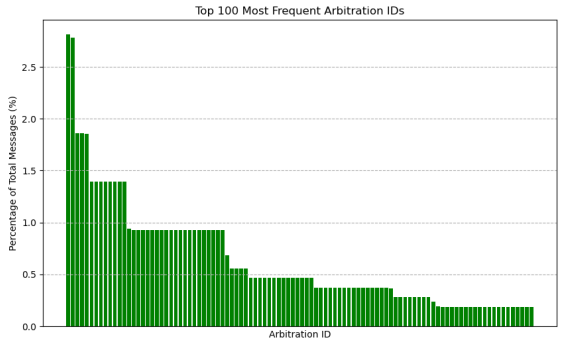


Figure 3.2: Top 100 most frequent Arbitration IDs

Additionally the cyclic time differences for each of the most frequent arbitration IDs are analyzed in order to find a realistic behavior of the timestamps of the messages that are to be replayed. Figure 3.3 shows the average cyclic time differences of the top five most frequent IDs, and those are the intervals in which the messages of each ID will be replayed.

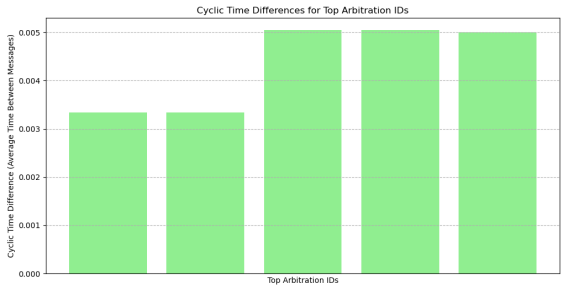


Figure 3.3: Cyclic time differences for Top 5 Arbitration IDs

For the Spoofing and Suspension attacks a set of DBC files have been used to

decode messages in the normal data. This is done with the Python library cantools. Using the decoded messages each message can be mapped to its transmitting ECU and consequently, every message transmitted from that ECU can be altered or removed in the normal data, resulting in the Spoofing and Suspension attacks. Only the unique IDs corresponding to an ECU are considered, excluding those IDs used by more than one ECU.

3.4 Generating attack data

Following the analysis, the tool proceeds to simulate synthetic attacks according to the identified patterns and support from the literature. Five types of attacks are considered i.e., Denial of Service, Fuzzy, Spoofing, Replay, and Suspension. These attacks span the four attack mechanisms outlined in Section 2.2, ensuring a wide range of potential threats.

3.4.1 Denial of Service (DoS)

The script designed to simulate a Denial of Service (DoS) attack creates a series of messages that mimic a flood of traffic on a CAN network. The functions simulate the DoS attack by generating bursts of high-priority messages during specified attack phases, known as intrusions, intended to overwhelm the network. The intrusions have a randomly chosen duration between 4 to 6 seconds and are separated by pauses, to reflect the unpredictability of an attack. The mean interval between messages is approximately 0.13 milliseconds, with pauses around a mean of 5 seconds, both subject to a standard deviation that introduces randomness into the timing of these messages. These values are variables and can be adjusted to add flexibility to simulate a wide range of DoS attacks, for example, if a higher interval between messages is wanted. The values chosen can be seen in Table 3.4 and are based on the analysis of the open source data to mimic those attack patterns. The calculation using the average intervals in the normal datasets to derive the chosen interval between messages can be seen in 3.2. The variability mimics real-world attack patterns and makes it harder for detection systems to predict and identify the attacks. Each message is created with a random payload, ensuring a variety of data within the attack pattern. Each message is assigned the highest priority ID, i.e. 0, and is integrated in intrusions with normal traffic data between two timestamps to create a dataset that reflects the network's state under attack conditions. This integration is performed with care to adjust timestamps and prevent overlaps to ensure that the attack messages are correctly distributed.

$$\begin{aligned} \text{DoS} &= \frac{1}{3}\left(A_{\text{KIA}} + A_{\text{SONATA}} + A_{\text{SPARK}}\right) \cdot 0.2 \\ &= \frac{1}{3} \cdot (0.6183 + 0.6059 + 0.6655) \cdot 0.2 \\ &\approx 0.13 \text{ ms} \end{aligned} \tag{3.2}$$

Table 3.4: DoS Attack Simulation Parameters

Parameter	**Value**
Intrusion Duration Range	4 to 6 s (randomly chosen between these values)
Pause Duration Mean	5 s (with variability)
Message Interval Mean	0.13 ms (with variability)
Standard Deviation for Pauses	$\frac{\text{Mean Pause}}{10}$ (Based on mean pause duration of 5 s)
Standard Deviation for Message Intervals	$\frac{\text{Mean Interval}}{3}$ (Based on mean message interval of 0.15 ms)
Arbitration ID Selection	0
Payload	Randomly generated data

3.4.2 Fuzzy Attack

The Fuzzy attack simulation begins by selecting a subset of message IDs from the normal traffic data, focusing on the most frequent ones to mimic legitimate traffic patterns. This selection is then used to generate a set of pseudo-random IDs by applying an XOR operation with a random number, expanding the variety of IDs in the simulated attack beyond what's typically observed but keeping the general pattern of how an ID looks in the normal messages. For each generated ID, random data payloads are created, simulating the content of each message. The messages are then merged with normal traffic data in intrusions, like in the DoS attacks described earlier. The intrusions last between 4 to 6 seconds (randomly chosen between those values) and with pauses of 5 seconds. The interval and pauses are both subject to a standard deviation that introduces randomness into the timing of these messages. The values chosen can be seen in Table 3.5 and are based on the analysis of the open source data to mimic those attack patterns. The calculation using the average interval in the normal datasets to derive the chosen interval between messages can be seen in 3.3. The timestamps are adjusted to avoid collisions, and to maintain a realistic traffic pattern the lower arbitration IDs are prioritized.

$$\begin{aligned} \text{Fuzzy} &= \frac{1}{3}\left(A_{\text{KIA}} + A_{\text{SONATA}} + A_{\text{SPARK}}\right) \cdot 0.2 \\ &= \frac{1}{3} \cdot (0.9746 + 1.0142 + 0.9334) \cdot 0.2 \\ &\approx 0.2 \text{ ms} \end{aligned} \tag{3.3}$$

Table 3.5: Fuzzy Attack Simulation Parameters

Parameter	**Value**
Intrusion Duration Range	4 to 6 s (randomly chosen between)
Pause Duration	5 s (with variability)
Message Interval Mean	0.2 ms (with variability)
Standard Deviation for Pauses	$\frac{\text{Mean Pause}}{10}$ (Based on mean pause duration of 5 s)
Standard Deviation for Message Intervals	$\frac{\text{Mean Interval}}{3}$ (Based on mean message interval of 0.2 ms)
Arbitration ID Selection	Pseudo-random IDs based on top 33% frequent IDs
Payload	Randomly generated data

3.4.3 Spoofing Attack

The Spoofing attack simulation identifies and removes messages from a specific ECU. It then generates new messages from that ECU, each with a pseudo-random payload, and inserts these messages in a pattern that mimics their normal transmission cycle. The pseudo-random payloads are generated similarly to the IDs in the Fuzzy attack, but instead of using the most frequent IDs, it uses the data in the messages sent by the specified ECU. To simulate a natural variation in the timestamps of the messages, a standard deviation is added. The simulation works by first identifying messages originating from the target ECU using a mapping of ECUs to their message IDs. This mapping is created by decoding the normal messages with a CAN Database file. By using IDs connected to a given ECU the simulation ensures that only messages relevant to the target ECU are considered for Spoofing. Once the relevant messages are identified, the payload of the message is replaced with pseudo-random data during a specific period of time. In this way, the simulation effectively creates a scenario where the target ECU appears to be sending out malicious data. Unlike the Fuzzy and DoS attacks, which launch their attacks in separate intrusions, the Spoofing attack continuously happens over a set period. The parameters and values can be seen in Table 3.6.

Table 3.6: Spoofing Attack Simulation Parameters

Parameter	**Value**
Target ECU	Specified ECU to spoof
Cyclic Behavior Simulation	Cyclic Behavior of normal messages
Standard Deviation for Cyclic Difference	1% of the base cyclic difference
Arbitration ID Selection	IDs of the targeted ECU
Payload	Pseudo-random based on original payload

3.4.4 Replay Attack

The Replay attack is simulated by first identifying specific messages that are likely targets for an attack, focusing on those IDs that occur frequently in the normal data. For each selected message, the average interval between its occurrences is calculated to understand its transmission pattern. This is done to ensure that the replayed messages are inserted in the normal data in a way that mimics their natural flow in the network. The Replay messages are generated by duplicating selected messages and adjusting their timestamps with the calculated intervals to maintain their cyclic pattern in the normal data. After the messages are created they are inserted into the normal data during a period of time. The parameters and values can be seen in Table 3.7.

Table 3.7: Replay Attack Simulation Parameters

Parameter	**Value**
Arbitration ID Selection	Top 5 most frequent message IDs
Payload	Original payload
Standard Deviation for Timing Variability	1% of the base cyclic difference

3.4.5 Suspension Attack

The simulation of the Suspension attack involves strategically removing messages from a specific target ECU within a specific time period, aiming to mimic the effect of the ECU being suspended from the network. This is done by first identifying the message IDs associated with a target ECU, using the same mapping of ECUs to their message IDs as in the Spoofing example. When the IDs have been identified the simulation filters out these messages from the normal data within the given time period. This mimics the suspension of the targeted ECU, creating a gap in the communication from that ECU.

3.5 Evaluation

The attack generator's performance was evaluated through the extraction of specific metrics from the datasets, utilizing Pandas and Numpy for the analysis. This process was conducted iteratively with the dataset creation to ensure that the proprietary data aligned with the open source data. The metrics identified as most relevant included the average ratio of attack messages to normal messages in each intrusion.

Further, several scattered plots were rendered from the open source data as well as the proprietary data, which shows the interval between all subsequent messages for normal data and attack data.

The results of the evaluation can be found in Chapter 5, Section 5.1.

Chapter 4
Intrusion Detection System

This chapter presents the methodology behind the implementation of two different Machine Learning (ML)-based Intrusion Detection Systems (IDS). These IDSs have been trained and tested on the synthetically created data described in Chapter 3. The chapter outlines the chosen machine learning models, together with their relevance and the motivation for inclusion in this project. This is followed by a detailed view of each model's architecture and the specific parameters involved. It then continues with describing the chosen features, the preprocessing of data, and the training process. Lastly, the methods of evaluation are discussed.

4.1 Selection of Models and Attacks

For the IDS implemented in this thesis, two types of deep learning models are evaluated, Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM), for their capabilities in handling CAN bus data. As motivated by Hossain et al. [35] the LSTM is a suitable choice for CAN messages as it is known for its efficacy in handling time-series data. The CNN, on the other hand, is noted for its strong feature extraction capabilities suitable for identifying distinct patterns in the data, including variations in CAN ID and payload.

Frequency-based IDSs can quite easily identify Fabrication and Suspension attacks by examining the timing and sequence of each ID [15]. In Masquerade attacks, one ECU is suspended while another impersonates it by mimicking its message pattern to send forged messages, leaving the frequency of message transmissions unchanged. Frequency-based detection systems fail to identify this type of attack because they do not change the typical frequency patterns of the network. Therefore, a method capable of analyzing the content within data frames becomes necessary. An LSTM model is particularly well-suited for this purpose due to its capability of sequence prediction problems. It makes it ideal for analyzing time-series data, such as CAN bus messages, which involve sequences of network traffic data. The LSTM model can learn to recognize the subtle changes in message content that indicate a Masquerade attack, even when message frequencies remain unchanged. The CNN can detect even minor deviations from established patterns, which is effective when subtle changes in

the payload occur.

This thesis focuses on supervised learning, where attacks such as Fabrication, Masquerade, and Replay mechanisms are chosen for training and evaluation. These attacks involve injecting messages into the CAN bus network. Since these attack files are labeled by message, they are suitable for supervised learning approaches. Suspension attacks, where messages are removed without injection, are not labeled and therefore not chosen for the supervised approach.

4.2 Data Preprocessing and Feature Extraction

The dataset used for training and testing the models includes the normal dataset provided by Scania, as detailed in Section 3.2.1, together with the synthetically created attack datasets. The attack datasets include DoS, Fuzzy, Spoofing, Suspension, and Replay attacks. The datasets are attributed as timestamp, CAN ID, DLC, data, and type. The 'type' attribute is labeled 'T' for the injected attack messages and 'R' for normal messages. From each dataset, 3 million messages were chosen, during the same time span, to make the model more effective. Table 4.1 presents the instances of each class in all of the datasets. The datasets, both sequences and single combined contain 85.8% of normal class instances and 6.6%, 4.8 %, 0.3%, 2.5% of DoS, Fuzzy, Spoofing, and Replay instances, respectively. The percentages of sequences for size 25 marked with the normal classification are 46.6% for Normal, and 15.8%, 15.1%, 7.1%, and 15.5% respectively for DoS, Fuzzy, Spoofing, and Replay attack classification in Table 4.2. For size 50, the percentages are 36.4% for Normal, and 15.8%, 15.1%, 12.6%, and 21.2% respectively for DoS, Fuzzy, Spoofing, and Replay. The datasets were combined and shuffled. The dataset is imbalanced reflecting real-world conditions where normal traffic outweighs anomalous traffic.

Table 4.1: Normal and attack instances

Attack	# of Messages	% of Total Msgs
Normal	10,290,642	85.8%
DoS	796,779	6.6%
Fuzzy	579,770	4.8%
Spoofing	38,131	0.3%
Replay	294,678	2.5%

Table 4.2: Normal and attack instances (Sequences)

Attack	# of Sequences for size 25 (Train/Val/Test)
Normal	3,910,820/838,055/837,733 (46.6%)
DoS	1,326,460/284,204/284,437 (15.8%)
Fuzzy	1,267,317/271,512/272,027 (15.1%)
Spoofing	592,972/127,345/127,027 (7.1%)
Replay	1,302,363/278,868/278,764 (15.5%)

Attack	# of Sequences for size 50 (Train/Val/Test)
Normal	2,971,032/636,650/636,651 (36.4%)
DoS	1,326,920/284,340/284,341 (15.8%)
Fuzzy	1,267,984/271,711/271,711 (15.1%)
Spoofing	1,057,144/226,531/226,531 (12.6%)
Replay	1,776,780/380,739/380,739 (21.2%)

From each dataset, the features extracted that contained adequate information were the CAN ID and the first eight bytes of the payload data. The CAN ID was transformed from hexadecimal to a binary format comprising 29 bits, with each bit being selected as a feature. Additionally, each byte of the payload data, initially in hexadecimal form, was converted to decimal and normalized using MinMaxScaler and chosen as a feature. These features were selected to reduce system complexity and execution time, as using a large number of data points in the classification could delay overall performance due to increased execution time [6]. Only the first 8 bytes were selected because choosing all 64 bytes would increase complexity, especially since the dataset consists of payloads where 90% are of an 8-byte size. The type label was converted to 0 for normal messages and 1 (DoS), 2 (Fuzzy), 3 (Spoofing), and 4 (Replay) for attack messages.

For both the LSTM and CNN models, two approaches were explored. The first approach treated each CAN message independently by considering its 37 features (comprising the CAN ID and payload) as a single-dimensional sequence. This method primarily focused on the feature-level analysis of individual messages. The second approach extended the analysis to handle sequences of messages to capture temporal dependencies across multiple messages. This approach processed the messages into overlapping sequences, using a sliding window mechanism that advanced one step at a time. Two different sequence lengths, 25 and 50, were empirically tested to determine the optimal configuration for anomaly detection. The sequences were labeled as 0 (Normal message) if there were no attack messages within that sequence and 1 (DoS), 2 (Fuzzy), 3 (Spoofing), or 4 (Replay) if there was at least one attack message. These sequences were split into training, validation, and testing data. Out of the data, 70 % was used for training, 15 % for validation, and the last 15 % was used as

testing data.

4.3 IDS Model Architecture

The architecture for the two types of models and their configurations can be seen in Table 4.3 and 4.4. The architecture was established with parameters inspired by the two IDS implementations presented in [35] and [36]. In those implementations, sigmoid was proven to be the most effective activation function and Nadam as the most effective optimizer, for both LSTM and CNN. A batch size of 256 or 512 was proven to be best, 256 was then chosen for the CNN model but for the LSTM model, 1024 had to be chosen due to computational time constraints. For the output activation function, softmax was chosen due to it being standard for multi-class classification tasks and similar for categorical_crossentropy for the loss function. The learning rate of 0.0001 was proven to be the most effective of the two referred implementations and therefore chosen for both LSTM and CNN. A filter size of either 256 or 512 for the CNN was recommended, thus 256 was chosen. These models were trained both sequence-wise having sequences of length 25 and 50 and pointwise i.e. a sequence of length 1.

Table 4.3: Parameter Values for 1D CNN Multiclass Classification

Parameters	Value
Sequence Length	1, 25 and 50
Activation Function	sigmoid
Filter size	256
Epoch	100
Output Layer Activation Function	softmax
Optimizer	Nadam
Batch Size	256
Learning Rate	0.0001
Loss Function	categorical_crossentropy

Table 4.4: Parameter Values for LSTM Multiclass Classification

Parameters	Value
Sequence Size	1, 25 and 50
Activation Function	sigmoid
Epoch	100
Output Layer Activation Function	softmax
Optimizer	Nadam
Batch Size	1024
Learning Rate	0.0001
Loss Function	categorical_crossentropy

4.4 Evaluation metrics

In this section, the performance measures for the methods are presented. A variety of metrics are utilized to measure the performance of the models. The detection accuracy shows how often a classification of a model is correct overall, as seen in Equation 4.1. When dealing with imbalanced datasets, relying only on accuracy is insufficient for assessing model performance. Precision, as seen in Equation 4.3, measures the accuracy of the attack predictions, i.e., how many of the positive predictions were true. A high precision means that when the IDS identifies an event as an attack, it is likely correct. Recall, as seen in Equation 4.2, measures the ability to detect all actual attacks, ensuring few or any attacks go unnoticed. The F1 score, as seen in Equation 4.4, provides a measure to balance precision and recall. It is an important metric for evaluating machine learning performance, especially with imbalanced datasets. The False Positive Rate (FPR), seen in Equation 4.5, is the proportion of negative instances that are incorrectly classified as positive and is an important measure when the cost of a false positive is high. The False Negative Rate (FNR), seen in Equation 4.6 is the proportion of positive instances that are incorrectly classified as negative and is an important measure if missing a positive instance is costly or dangerous. In the context of anomaly detection, it is especially important to have a low FNR for the *normal class*, as generating an excessive number of false alarms, where normal traffic is incorrectly classified as malicious can overwhelm security analysis system (e.g., the cyber security management system), leading to alert fatigue and making it difficult to respond to real threats among the noise of false positives.

To provide a visual summary of the IDS performance, a confusion matrix will be included in the analysis. This matrix offers a clear visualization of how the model performs concerning each class by displaying the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

$$\text{Detection Accuracy (Acc)} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$\text{Recall (Rec)} = \frac{TP}{TP + FN} \tag{4.2}$$

$$\text{Precision (Pre)} = \frac{TP}{TP + FP} \tag{4.3}$$

$$\text{F1 Score (F1)} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{TN + FP} \tag{4.5}$$

$$\text{False Negative Rate (FNR)} = \frac{FN}{TP + FN} \tag{4.6}$$

The results of the IDS performance can be found in Chapter 5, Section 5.2.

Chapter 5
Results

This chapter presents the results obtained in this thesis. The first part focuses on the results of the synthetically generated attack files and consequently the evaluation of the attack generator as a whole, presenting the calculated metrics and plots for the proprietary as well as open source data. The second part focuses on the results of the evaluation of the implemented ML-driven IDS, i.e., utilizing the Convolutional Neural Network (CNN) and the Long Short-Term Memory (LSTM) models.

5.1 Synthetic Attack Generation

This section presents the results of the synthetically created attack files. These results include the average ratio of attack messages in intrusions, complemented by visual plots from both proprietary and open-source data.

The outcomes for the average interval metric are presented in Table 5.1. These metrics were extracted from the Fuzzy and DoS data and from both proprietary and open source data. The results show that the interval between the messages in the proprietary data is around 0.07 ms while the interval in the open source data is larger and around 0.4 ms.

Table 5.1: Average interval between every message in DoS and Fuzzy datasets

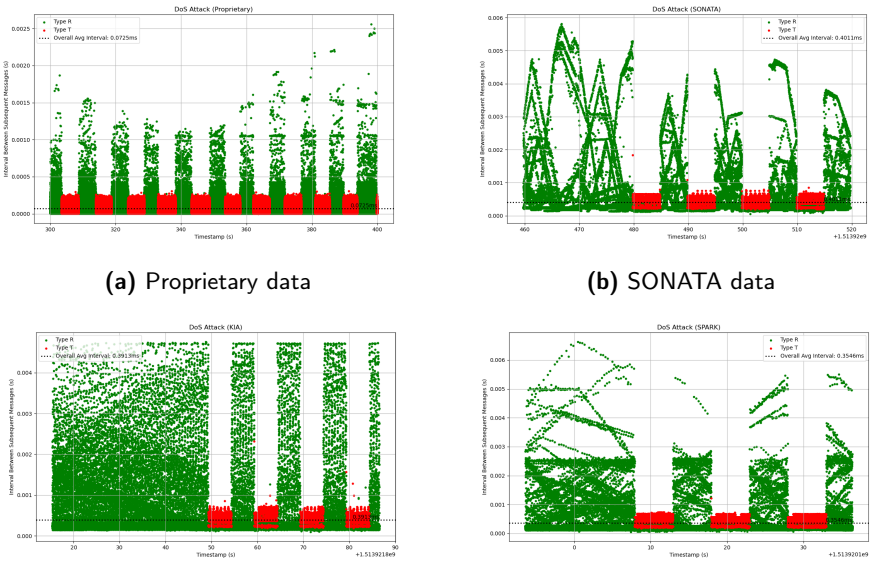
	Proprietary	SONATA	KIA	SPARK
DoS (ms)	0.0725	0.4011	0.3913	0.3546
Fuzzy (ms)	0.0773	0.4438	0.4016	0.3961

The outcomes for the average ratio metric are presented in Table 5.2. These metrics were extracted from the Fuzzy and DoS data and from both proprietary and open source data. The average ratio of messages within an intrusion, i.e., the percentage of attack messages are of similar values between all datasets. For the proprietary data and the DoS attack the percentage of attack messages is 38.6 % and for the open source data and the DoS attack the percentage is between 35.6 - 39.3 %. For the Fuzzy attack, the percentage is around 28 % for all four datasets.

Table 5.2: Average ratio in intrusion (nbr of T/ nbr of R and T)

	Proprietary	SONATA	KIA	SPARK
DoS (%)	38.610	39.269	35.645	35.645
Fuzzy (%)	28.293	28.565	28.170	28.170

The plots presenting the interval between every subsequent message in the DoS and Fuzzy attack files, for proprietary and open source data can be found in 5.1, and 5.2. The red represents the attack messages (labeled T) while the green represents the normal messages (labeled R), i.e., each attack interval corresponds to the red sections of the plot. The plots show that an attack leads to a decrease in the interval between messages in all four files. In each plot, the average interval between all messages is shown as a dotted line for clarification. For the proprietary data, a portion of the file has been plotted (100 seconds), which corresponds to the total time of the open source data files. The x-axis represents the total timestamp of each message and the y-axis the interval.



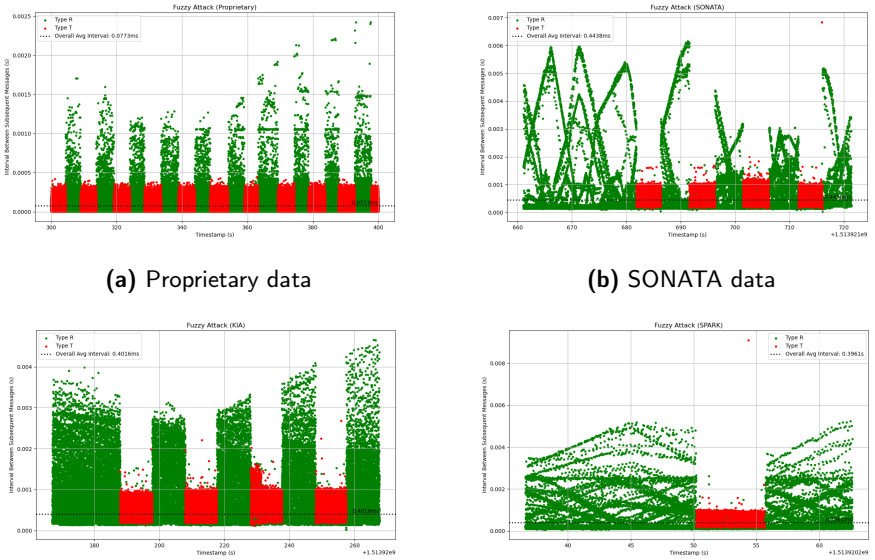
(a) Proprietary data

(b) SONATA data

(c) KIA data

(d) SPARK data

Figure 5.1: DoS: The interval between subsequent messages (s) on the y-axis and timestamp (s) on the x-axis. Attack messages (T) are marked in red and normal messages (R) are marked in green



(a) Proprietary data

(b) SONATA data

(c) KIA data

(d) SPARK data

Figure 5.2: Fuzzy: The interval between subsequent messages (s) on the y-axis and timestamp (s) on the x-axis. Attack messages (T) are marked in red and normal messages (R) are marked in green

Additionally, the plot in Figure 5.3 presents the interval between every subsequent message transmitted from the targeted ECU in a Suspension attack. Thus, the intervals plotted are all between messages sent by the targeted ECU. The gap in the plot shows where the ECU has been suspended for about 200 seconds and it is clear that there are no intervals plotted here, i.e., there are no messages sent during this period of time.

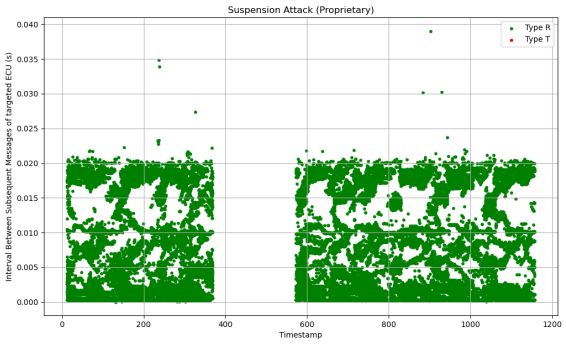


Figure 5.3: The interval between subsequent messages for the Suspension attack, filtered on the targeted ECU

Figure 5.4 presents the interval between every subsequent message transmitted from the targeted ECU during a Spoofing attack, the red dots represent the injected (spoofed) messages, and the green dots correspond to all other normal messages. For comparison, Figure 5.5 displays data from the normal log file for the same time frame, where the blue dots indicate messages from the targeted ECU, and the green dots represent all other messages on the bus.

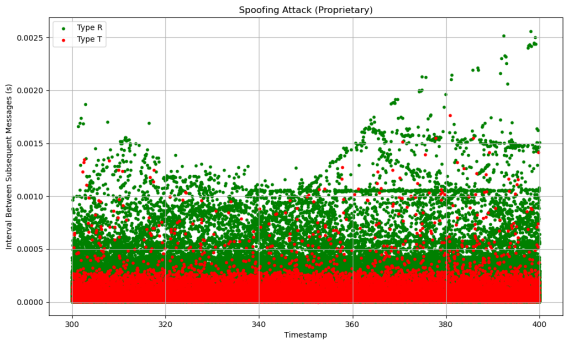


Figure 5.4: The interval between subsequent messages for the Spoofing attack

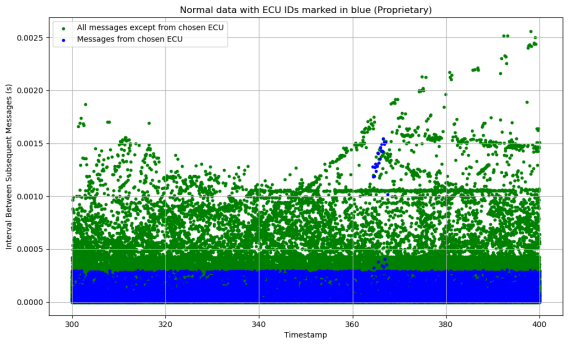


Figure 5.5: The interval between subsequent messages in normal data within the time frame for the Spoofing attack seen in Figure 5.4

Lastly, Figure 5.6 presents a small slice of the dataset during a Replay attack, showing how the messages are being replayed. ID 1 through 5 are the top 5 most occurring messages in the file and those targeted for the Replay attack. The green crosses represent normal messages being sent and the red crosses are those same messages being replayed.

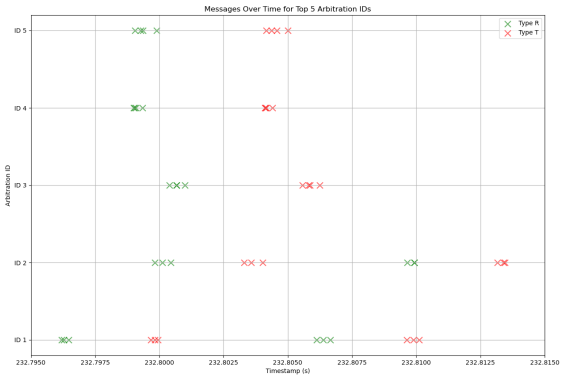


Figure 5.6: Normal and Replayed messages during a small timeframe

5.2 Intrusion Detection System

This section presents the evaluation results of the implemented models for the IDSs. The chosen metrics to evaluate the model's performance were accuracy, recall, false positive rate (FPR), false negative rate (FNR), and F1 score. These are motivated in Chapter 4.

5.2.1 LSTM models

Table 5.3 presents the results for the LSTM model trained on sequences of length 1, i.e., trained on individual messages (pointwise). The overall accuracy for the model was 97.52 %. This model was unable to detect any instances of Replay, and therefore the model shows very poor results for this class. The result shows that the model detects instances of DoS very well with perfect scores in all metrics. Fuzzy can also easily be detected with high values in precision, recall, and F1 score and low values for FPR. The Spoofing class is slightly more difficult to detect with a precision of 0.9797 and also a slightly lower recall and F1 score. The Normal class has the highest FPR of 0.1733 meaning that about 17% of all messages classified as normal were actually attack messages. The FNR was low for the normal class 0.0001, meaning that nearly 0.01% of the normal messages were classified as attack messages.

The confusion matrix (for the LSTM model trained on sequences of length 1) in Figure 5.7 shows that most messages are located in the diagonal, i.e. classified to their correct class. However, all Replay classes have been classified as Normal messages. All messages of the DoS class are correctly classified as DoS. For Fuzzy and Spoofing some messages are classified as Normal and some messages of class Normal are classified as Fuzzy and Spoofing, a few messages of Fuzzy have been classified as Spoofing.

Table 5.3: Results for LSTM (without sequences)

Attack	Accuracy	Precision	Recall	F1 Score	FPR	FNR
Normal		0.9720	0.9999	0.9858	0.1733	0.0001
DoS		1.0000	1.0000	1.0000	0.0000	0.0000
Fuzzy	**97.52 %**	0.9995	0.9976	0.9985	0.0000	0.0024
Spoofing		0.9797	0.9739	0.9767	0.0001	0.0261
Replay		0.0000	0.0000	0.0000	0.0000	1.0000

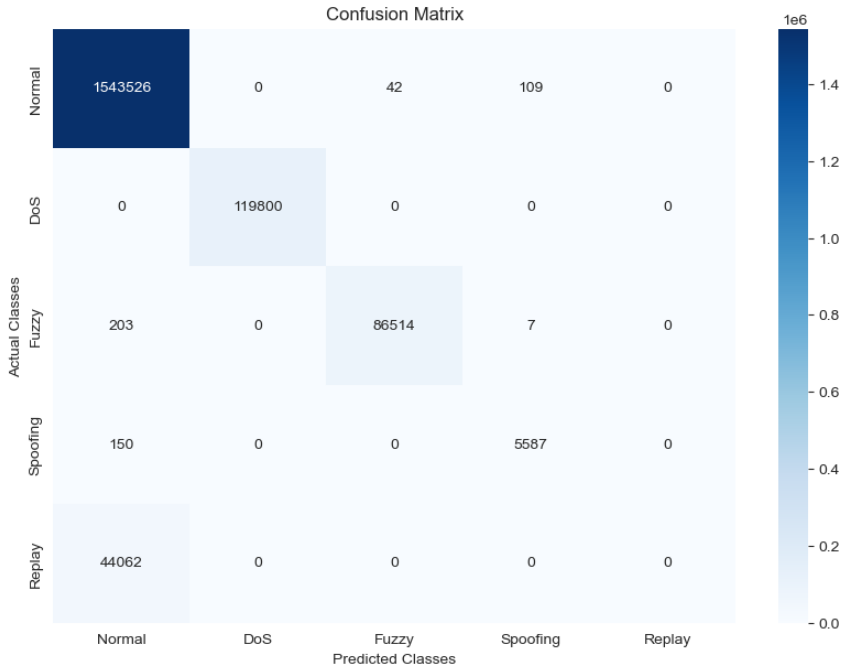


Figure 5.7: LSTM Confusion Matrix for Table 5.3 (without sequences)

Table 5.4 presents the result for the LSTM model trained on sequences of length 25. The overall accuracy for this model was 99.49 %. The model performs very well on the DoS and Fuzzy classes with high scores on precision, recall, and F1 scores and low scores on low scores on FPR and FNR. The FNR for the normal class was 0.0037, meaning that about 0.37 % of all normal messages were wrongly classified as attack messages. The FPR of the normal class was 0.0063, which shows that about 0.63 % of all attack messages were wrongly classified as normal messages.

The confusion matrix (of the LSTM model trained on sequences of length 25) in Figure 5.8 shows that most instances are located in the diagonal, classified to their correct class. About 2000-3000 sequences of classes Spoofing and Replay have been wrongly classified as instances of the Normal class, and about 3000 sequences of the Normal class have been classified as instances of the Replay class. Some sequences of DoS and Fuzzy are incorrectly classified as instances of Normal, Spoofing, and Replay.

Table 5.4: Results for LSTM (sequences of 25)

Attack	Accuracy	Precision	Recall	F1 Score	FPR	FNR
Normal		0.9928	0.9963	0.9945	0.0063	0.0037
DoS		1.0000	0.9999	1.0000	0.0000	0.0001
Fuzzy	**99.49 %**	0.9999	0.9994	0.9997	0.0000	0.0006
Spoofing		0.9968	0.9839	0.9903	0.0002	0.0161
Replay		0.9899	0.9861	0.9880	0.0018	0.0139

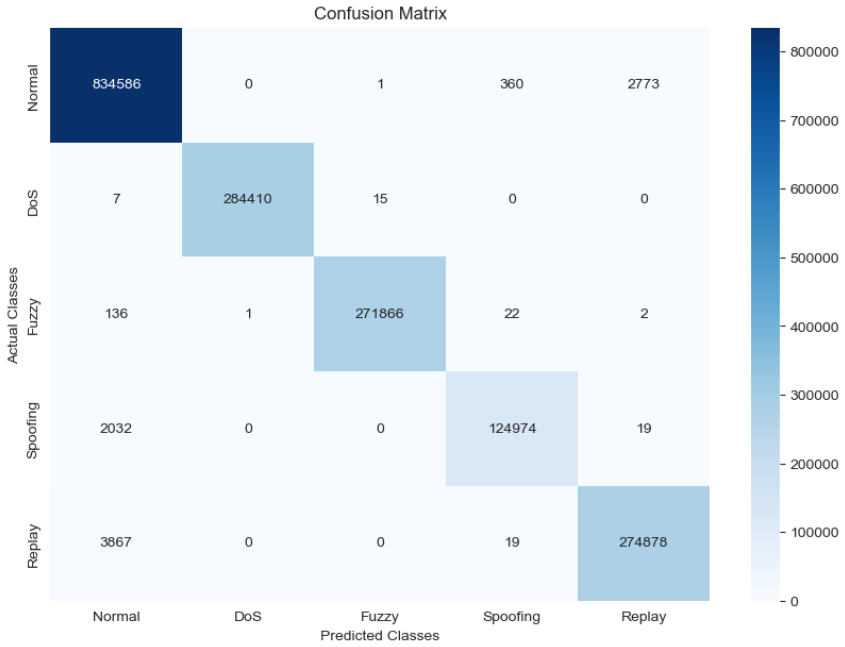


Figure 5.8: LSTM Confusion Matrix for Table 5.4 (sequences of size 25)

Table 5.5 presents the result for the LSTM model trained on sequences of length 50. The overall accuracy for this model was 99.84 %. The model performs best on the DoS and Fuzzy classes with high scores on precision, recall, and F1 scores and low scores on FPR and FNR. The precision for the Spoofing class is also high but the recall and F1 score is lower than DoS and Fuzzy. The Replay class has a lower precision than Spoofing as well as a lower recall and F1 score. The FNR for the normal class was 0.0009, meaning that about 0.09 % of all normal messages were wrongly classified as attack messages. The FPR of the normal class was 0.0019, which shows that about 0.19 % of all attack messages were wrongly classified as normal messages.

Figure 5.9 presents the confusion matrix for the LSTM model trained on

sequences of length 50. The majority of instances are predicted in the diagonal i.e. as the correct class. Few instances of DoS and Fuzzy have been wrongly classified summing up to barely 50 instances. The Spoofing class has a few more misclassifications of about 400, however, Replay is worse with around 1800 misclassifications. Almost 570 instances of the Normal class have been classified as attack instances.

Table 5.5: Results for LSTM (sequences of 50)

Attack	Accuracy	Precision	Recall	F1 Score	FPR	FNR
Normal		0.9965	0.9991	0.9978	0.0019	0.0009
DoS		1.0000	0.9999	1.0000	0.0000	0.0001
Fuzzy	**99.84 %**	1.0000	0.9999	0.9999	0.0000	0.0001
Spoofing		0.9998	0.9981	0.9990	0.0000	0.0019
Replay		0.9986	0.9953	0.9970	0.0004	0.0047

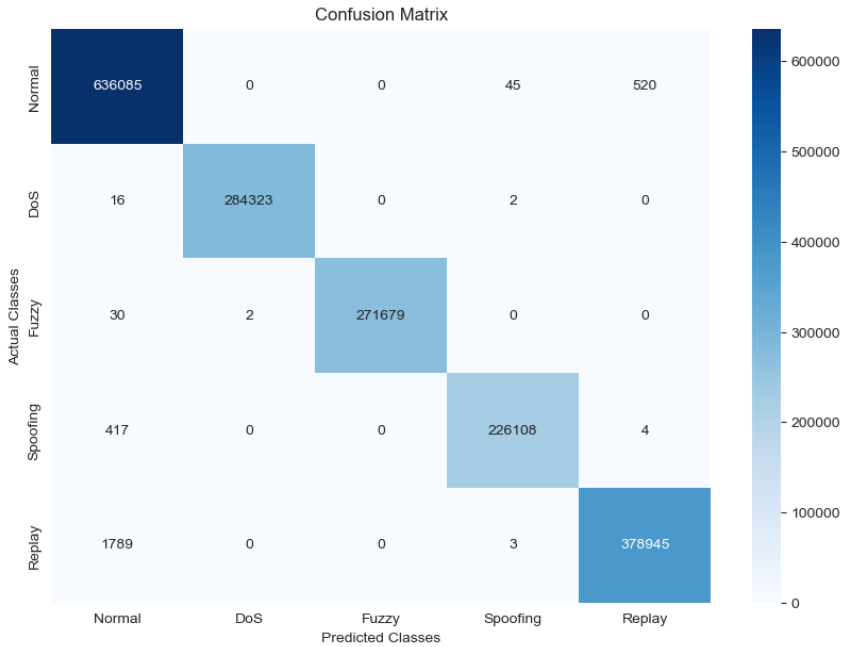


Figure 5.9: LSTM Confusion Matrix for Table 5.5 (sequences of size 50)

5.2.2 CNN models

Table 5.6 presents the results for the CNN model trained on sequences of length 1, i.e., trained on individual messages (pointwise). The overall accuracy for the model was 97.55 %. This model was unable to detect any instances of Replay, and therefore the model shows poor results for this class. The result shows that the model detects instances of DoS very well with perfect scores in all metrics. Fuzzy and Spoofing can also easily be detected with high values in precision, recall, and F1 score and low values for FPR.The Normal class has the highest FPR of 0.1724 meaning that about 17% of all messages classified as normal were actually attack messages. The FNR was low for the normal class 0.0000, meaning that nearly 0% of the normal messages were classified as attack messages.

Figure 5.10 displays the confusion matrix for the CNN model trained on sequences of length 1. It shows that most messages are located in the diagonal, classified to their correct class. However, all messages of class Replay have been classified as Normal messages. All messages of the DoS class are correctly classified as DoS. For Fuzzy and Spoofing some messages are classified as Normal and some messages of class Normal are classified as Fuzzy and Spoofing, a few messages of Fuzzy have been classified as Spoofing.

Table 5.6: Results for 1D CNN (without sequences)

Attack	Accuracy	Precision	Recall	F1 Score	FPR	FNR
Normal		0.9722	1.0000	0.9859	0.1724	0.0000
DoS		1.0000	1.0000	1.0000	0.0000	0.0000
Fuzzy	**97.55 %**	0.9990	0.9995	0.9993	0.0000	0.0010
Spoofing		0.9988	0.9941	0.9964	0.0000	0.0059
Replay		0.0000	0.0000	0.0000	0.0000	1.0000

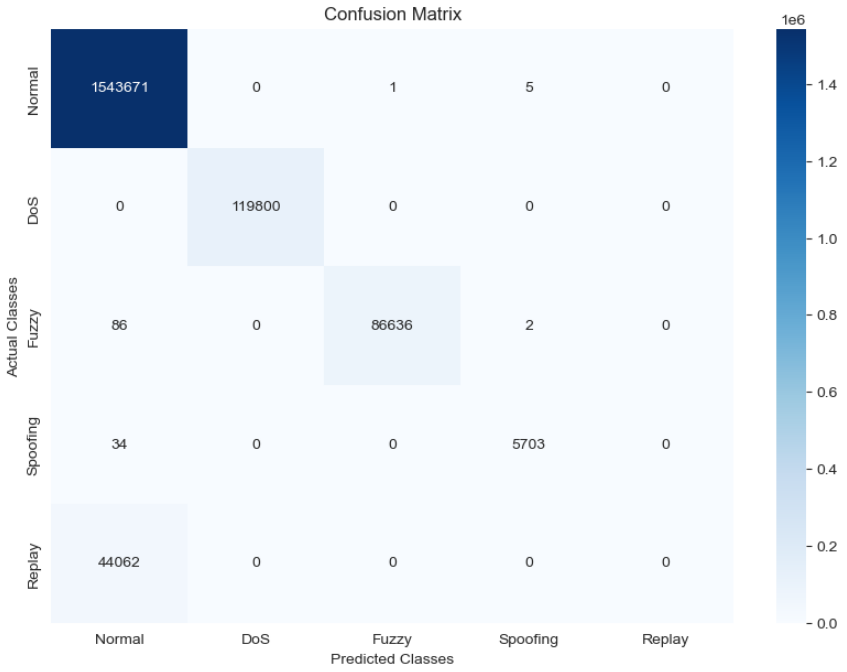


Figure 5.10: 1D CNN Confusion Matrix for Table 5.6 (without sequences)

Table 5.7 presents the result for the CNN model trained on sequences of length 25. The overall accuracy for this model was 99.44 %. The model performs best on the DoS and Fuzzy classes with high scores on precision, recall, and F1 scores and low scores on FPR and FNR. The model is worse at classifying instances of the Normal, Spoofing, and Replay classes with Replay having the lowest scores. The FNR for the normal class was 0.0036, meaning that about 0.36 % of all normal messages were wrongly classified as attack messages. The FPR of the normal class was 0.0074, which shows that about 0.74 % of all attack messages were wrongly classified as normal messages.

The confusion matrix (of the CNN model trained on sequences of length 25) presented in Figure 5.11 shows that the majority of sequences have been classified to their correct class as they are located in the diagonal of the matrix. Approximately 3000 sequences of the Spoofing class and 4000 of the Replay class have been incorrectly classified as Normal. Very few sequences of DoS are incorrectly classified while some sequences of Fuzzy are classified as Normal and DoS. Almost 3000 sequences of class Normal have been classified as either DoS, Fuzzy, Spoofing, or Replay.

Table 5.7: Results for 1D CNN (sequences of 25)

Attack	Accuracy	Precision	Recall	F1 Score	FPR	FNR
Normal		0.9916	0.9964	0.9940	0.0074	0.0036
DoS		1.0000	1.0000	1.0000	0.0000	0.0000
Fuzzy	**99.44 %**	0.9998	0.9998	0.9998	0.0000	0.0002
Spoofing		0.9997	0.9761	0.9877	0.0000	0.024
Replay		0.9895	0.9856	0.9876	0.0019	0.0144

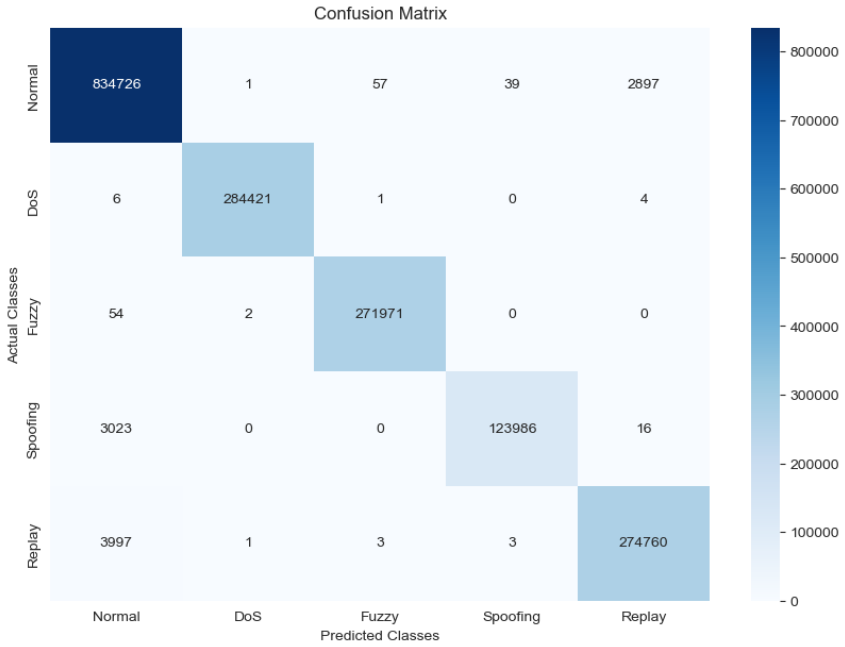


Figure 5.11: 1D CNN Confusion Matrix for Table 5.7 (sequences of size 25)

Table 5.8 presents the result for the CNN model trained on sequences of length 50. The overall accuracy for this model was 99.72 %. The model performs best on the DoS and Fuzzy classes with high scores on precision, recall, and F1 scores and low scores on FPR and FNR. The precision for the Spoofing class is also high but the recall and F1 score is lower than DoS and Fuzzy. The Replay class has a lower precision than Spoofing but a higher recall and F1 score. The FNR for the normal class was 0.0001, meaning that about 0.1 % of all normal messages were wrongly classified as attack messages. The FPR of the normal class was 0.0042, which shows that about 0.42 % of all attack messages were wrongly classified as normal messages.

Figure 5.12 illustrates the confusion matrix for the CNN model with sequences of

length 50. This shows that the majority of sequences are correctly classified, and located in the diagonal of the matrix. Barely 100 sequences of the Normal class have been classified as being an attack sequence and only 1 sequence of DoS is incorrectly classified. The Fuzzy and Spoofing class also shows few misclassifications while about 4600 instances of Replay have been misclassified as Normal instances.

Table 5.8: Results for 1D CNN (sequences of 50)

Attack	Accuracy	Precision	Recall	F1 Score	FPR	FNR
Normal		0.9924	0.9999	0.9961	0.0042	0.0001
DoS		1.0000	1.0000	1.0000	0.0000	0.0000
Fuzzy	**99.72 %**	1.0000	0.9999	0.9999	0.0000	0.0001
Spoofing		0.9999	0.9991	0.9995	0.0000	0.0009
Replay		0.9998	0.9878	0.9938	0.0000	0.0122

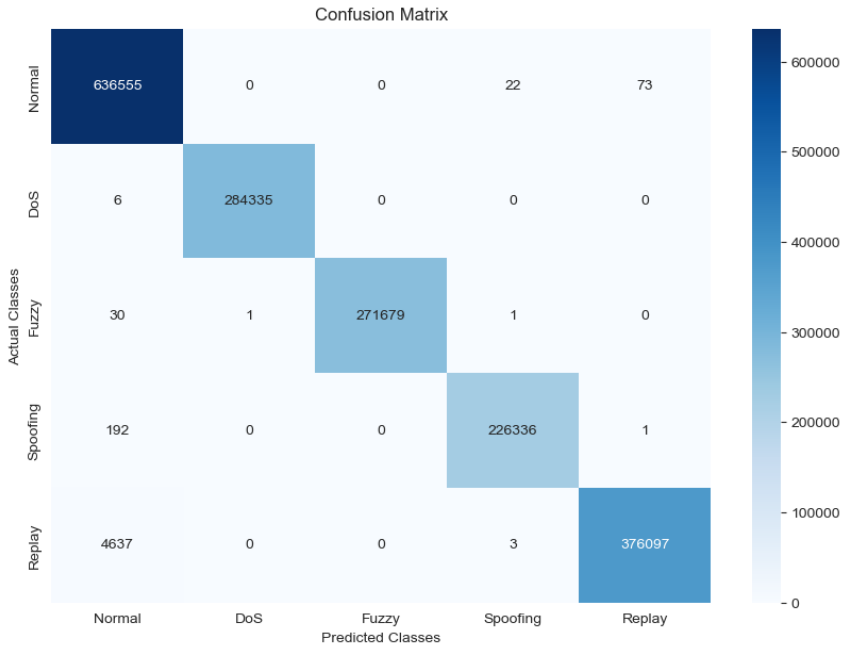


Figure 5.12: 1D CNN Confusion Matrix for Table 5.8 (sequences of size 50)

5.2.3 Comparative analysis of LSTM and CNN models

To summarize how the LSTM and CNN models adapt to varying sequence lengths, two important performance metrics, i.e., accuracy and FNR on the Normal class are highlighted. Table 5.9 shows the overall accuracy of each model at different sequence lengths, indicating that both models generally improve as the sequence length increases. The LSTM models had a higher accuracy than the CNN model with the sequence-based training while the CNN model was slightly better than the LSTM model with the point-based training. The LSTM model with sequences of length 50 had the best accuracy of 99.84 %. The model with the lowest accuracy was the LSTM model with no sequences which had an accuracy of 97.52 %.

Table 5.9: Comparison of LSTM and CNN Model Overall Accuracy by Sequence Length

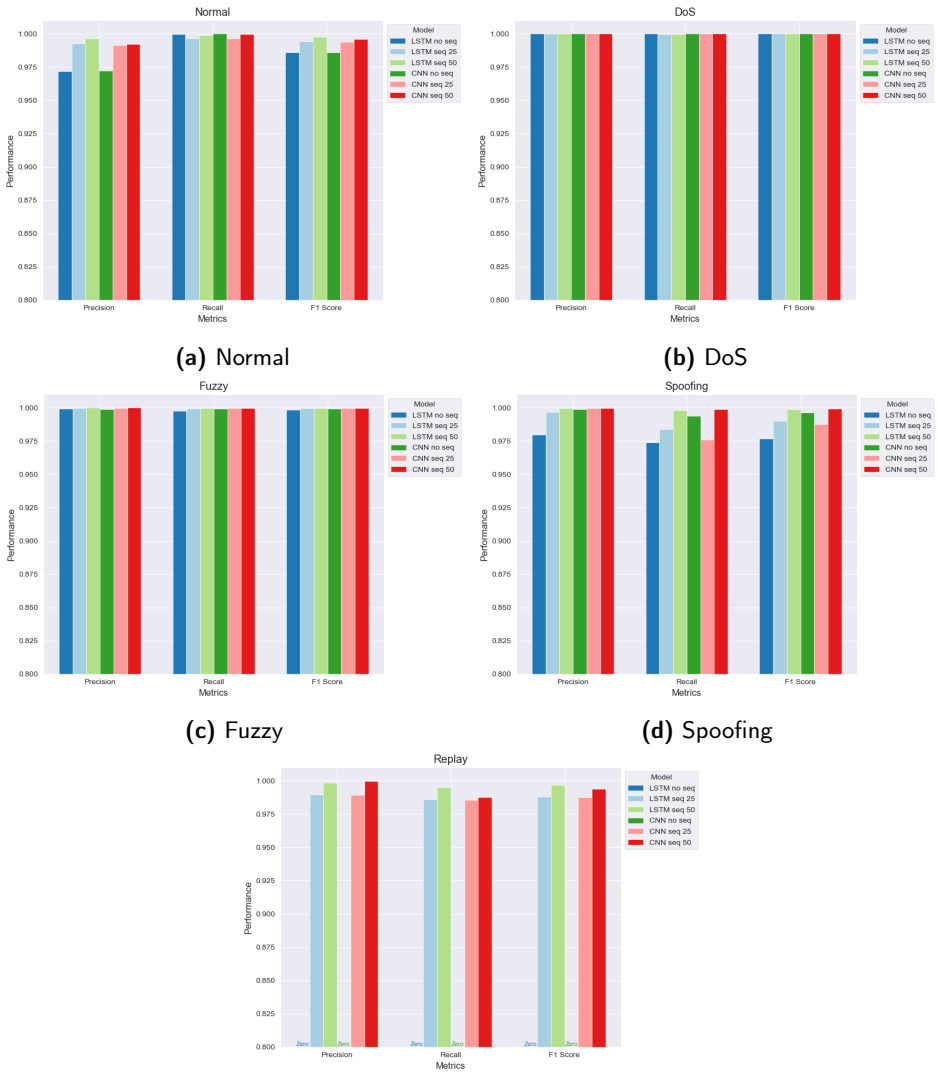
Sequence Length	LSTM Accuracy	CNN Accuracy
Pointwise	97.52 %	97.55 %
25	99.49 %	99.44 %
50	**99.84** %	99.72 %

Table 5.10 focuses on the False Negative Rate (FNR) for normal data. The FNR for the CNN models was lower than the LSTM models in all three cases. The models with sequence length 25 had the highest FNR.

Table 5.10: Comparison of FNR for LSTM and CNN Models on Normal Data by Sequence Length

Sequence Length	LSTM FNR	CNN FNR
Pointwise	0.01 %	**0.00 %**
25	0.37 %	0.36 %
50	0.09 %	0.01 %

Figure 5.13 presents five diagrams showing the overall results for each class and the metrics precision, recall, and F1 score for each of the six model configurations. The diagrams are plotted from 0.8 to 1.0 to get a more clear visualization of the differences. These diagrams show that all models perform well on Fuzzy and DoS with scores close to 1 in all three metrics. The models generally have a high recall on the Normal class but lower values in precision and F1 score. Regarding the Spoofing class, the models have a higher precision than the Normal class but a lower recall and F1 score. The diagram presenting the results for the Replay class shows that the metrics for the models trained on sequences of length 1 is 0, and accordingly for this attack the trained models perform generally worse than other attacks.



(a) Normal

(b) DoS

(c) Fuzzy

(d) Spoofing

(e) Replay

Figure 5.13: Performance metrics (precision, recall and F1 score) for each class and each model

Chapter 6
Discussion

This chapter discusses the results of the attack generator and the machine learning-based IDS used for detecting attacks on the CAN bus data. It explores the implications, addresses the limitations, and provides suggestions for future research.

6.1 Attack Generator

This section elaborates on the validity of the synthetic data produced and the analysis of specific attack mechanisms.

6.1.1 Validity of synthetic data

Synthetically generated attack data, while useful for training and testing an IDS, presents significant challenges. One major issue is the difficulty in verifying the real impact of these attacks on a vehicle's operational behavior. Unlike real attacks, where the direct effects can be observed and measured on actual vehicle systems, the synthetic attacks in this thesis can not. This means that while synthetic data can simulate scenarios, it cannot fully represent what would happen in a real-world setting. This limitation is due to predicting the specific impacts of these injected messages on a real vehicle's systems is too challenging for the scope of this thesis. Additionally, the normal data on which the attack data was based had been collected from a single driving session, limiting the data's variability and diversity.

6.1.2 Analysis of the generated attack files

The synthetic attack data for the Fabrication mechanism was benchmarked against the open source datasets, demonstrating a similar average ratio of attack messages in intrusions, as shown in Table 5.2. This similarity is important for the effectiveness of the IDS, as it leads the IDS to be tested under conditions that resemble actual attack scenarios to some extent. For Suspension, Masquerade, and Replay attacks, the dependency on injection frequency is not as critical as it is for the Fabrication attack mechanism. The effectiveness of the Suspension attack mechanism is straightforward to verify, as it involves removing

messages from a specific Electronic Control Unit (ECU). This can be observed in Figure 5.3, which shows the absence of messages from the targeted ECU. For the Replay attack, verification is similarly direct, where Figure 5.6 demonstrates that the messages are being replayed as expected. For the Masquerade attack mechanism, the Spoofing attack is displayed in Figure 5.4 and the actual normal activity during the same time frame before the injected attack is shown in Figure 5.5. In the Spoofing plot, the injected messages are scattered throughout and tend to have a wider distribution of intervals between messages, some overlapping with the normal intervals and some with a noticeably larger gap compared to the normal activity. The spoofed messages (red) in the attack plot show more timing variability, indicating a Masquerade attack closer to the reality. This is consistent with realistic scenarios where it is challenging for an attacker to duplicate the precise timing of the normal messages.

6.1.3 Attack patterns and open source data

In analyzing the impact of attacks on in-vehicle systems, it is important to consider the different vehicle types used within the datasets. The open source data included three different vehicles, each showing distinct responses to the attacks. This variation is demonstrated in Figures 5.1 and 5.2, which show differences in bus load and message frequency distribution across these vehicles. Moreover, the proprietary dataset differs not only in vehicle type, being sourced from a truck as opposed to a car, but also in the communication protocol used. This dataset includes CAN-FD messages, whereas the open source data is limited to the standard CAN protocol. CAN-FD's increased payload capacity and higher data rate could influence the impact of attacks compared to those on standard CAN systems.

This thesis focuses on generating attack patterns that are important for an IDS to detect, particularly because they could severely impair in-vehicle functions. As discussed, the synthetically simulated attacks in the generated files do not predict or include the actual effects on vehicle operation. Nonetheless, it's important to ensure that the simulated attacks would realistically affect a vehicle. To implement attacks that can severely impair in-vehicle function, an analysis of open source datasets was used to determine attack frequency and type of messages to mimic a realistic performed attack for the Fabrication mechanism. The open source data describe in their paper that the Fuzzy attack successfully resulted in malfunctions such as unintended activation of vehicle components. Yet, for the DoS attack no malfunction was mentioned, this absence in the available data suggests a potential gap in understanding the full impact of such attacks on the CAN bus.

Furthermore, the attacks in the open source datasets are limited to approximately 1 minute which does not sufficiently represent the range of conditions and durations that vehicles may encounter in real-world scenarios. This limited duration may not capture the long-term effects of attacks or the vehicle's response over an extended period. Extending the simulation time could

provide a more comprehensive understanding of the effects of long or continuous attacks but for the scope of this thesis, the patterns of attacks are sufficient to generate realistic attack data.

6.2 Intrusion Detection System

This section discusses the results of the Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) models. Moreover, it addresses the limitations of the implementation of the IDSs.

6.2.1 Observations from results

The results from testing the IDS, implemented with both LSTM and CNN models, demonstrated different insights. For example, a high accuracy in detecting DoS and Fuzzy attacks for both models. These attacks typically present clear deviations from normal traffic patterns, which the models could successfully identify and classify. The DoS attack involves injecting messages with a high-priority arbitration ID, in this case, ID 0. Since this ID deviates significantly from those of normal messages, the attack can be easily detected. The normal messages do not use an arbitration ID of 0, which enables the model to identify the attack messages with high probability. Additionally, the models are also highly capable of detecting Fuzzy attacks, which involve injecting messages of random arbitration IDs that, although random, still fall within the range of normal message IDs. This capability demonstrates the model's effectiveness in identifying a variety of attacks characterized by slight deviations from the standard range of arbitration IDs.

The LSTM and CNN models trained with input sequence lengths of 25 and 50 showed variations in their detection accuracy. Specifically, shorter sequences or pointwise tended to be sufficient for detecting straightforward attack patterns such as DoS and Fuzzy. However, for more complex scenarios like Spoofing and Replay attacks, extending the sequence length to 50 generally improved detection capabilities, as shown by the tables in section 5.2. The IDS, both for LSTM and CNN, only detects Replay attacks when using sequences as input. Replay attacks, the retransmission of valid messages, are challenging to identify without a sequence-based model as it need to understand temporal dependencies between messages. The Replay attack is detected with both sequences of 25 and 50, as mentioned with slightly better performance for the longer sequence in accuracy. While the Spoofing attack is detected with better performance for models with sequences as input, the models with points as input could still detect them. Since the Spoofing attack uses spoofed IDs i.e. IDs already existing in the normal data together with a random payload, this shows that the model is adept at detecting deviations in the payload as well.

The metric identified as one of the most important is the FNR of normal data, which helps minimize false alarms. As indicated in Table 5.10, the FNR for LSTM and CNN are very low at 0.01% and 0.00% respectively for pointwise

input. With longer sequences, these rates increase, which is due to the model's ability to detect Replay messages, with most misclassifications occurring in this category. The CNN models had a slightly lower FNR for the Normal class than the LSTM models for all model configurations, i.e. fewer instances of the Normal class are misclassified by the CNN models. However, the FPR is generally higher for the CNN models, meaning that more attack instances are missed and classified as Normal. The FNR is the highest when the models are trained on sequences of length 25, which can be explained by the difficulty in classifying Spoofing and Replay. With a longer sequence length, more instances of Spoofing and Replay can be correctly classified and the FNR becomes lower. This indicates that a longer sequence length is better for detecting these types of attacks.

The results showed that the LSTM models performed slightly better than CNN when trained on sequences, which can be due to the LSTM being particularly skilled at handling sequential data. While CNNs are great at spatial pattern recognition, such as in image processing, they can be less efficient at finding the long-term dependencies needed for sequence processing. LSTMs that are designed to learn information across time steps, can therefore be a better match for tasks where historical context influences future outcomes, such as CAN data. The model that performed best across all classes in terms of accuracy, precision, recall, and F1 score was the LSTM model trained on sequences of length 50. However, if the goal is to have a high accuracy together with a low FNR the CNN trained on sequences of 50 might be the better option.

6.2.2 Limitations

Supervised Learning

Using only supervised learning limits the model's detection ability to the attacks that it is trained on. The models proposed in this thesis are only trained on four types of attacks, which means that it is only able to detect these four types of attacks. Even though these four attacks are relevant to this topic, there could be more attacks not yet known or discovered in the literature. Thus, only using supervised learning for this problem is a limitation, since this means that the IDS will be unable to detect unknown attacks.

Dependency on Synthetic Data

As discussed earlier, the use of synthetically generated data for training and testing also poses a limitation. There is a big risk that the models may not perform as well when exposed to real-world attack scenarios that weren't adequately represented in the training data. The synthetically created data does not contain the same amount of randomness as real scenario data, therefore the patterns for each attack might be easier to learn for the models.

CAN-FD

The models use the payload as an input feature, and for reduced complexity, only the first 8 bytes were considered. However, this approach presents limitations, particularly because the log file contains CAN-FD messages with payload sizes of up to 64 bytes. While the majority of messages in the log file are of size 8, using only the first 8 bytes could potentially result in the loss of some valuable data contained within the extended payload. Although the models achieve high accuracy, considering the entire payload or enhanced feature extraction might improve the results.

6.3 Further research

The research in this thesis has investigated creating synthetic attack data and detecting these attack patterns with deep neural networks. However, there are areas for further exploration, particularly in addressing challenges with synthetic data and supervised learning models.

Generating realistic attack data

While this thesis created a tool for generating synthetic data, there is a need for more advanced simulation tools that can create highly realistic attack scenarios in different driving conditions. Currently, the normal data on which the attack data was based, had been derived from a single driving session. Future research should focus on enhancing synthetic datasets with data from multiple driving sessions across diverse conditions. This approach would increase the diversity of normal data, thereby improving the generalizability of the IDS. By expanding the dataset, the models trained on this data can better adapt to different driving behaviors and conditions, improving their ability to detect and respond to cyber threats in real-world scenarios. Alternatively, despite being expensive and potentially risky, getting extensive data from real attacks performed on an actual driving vehicle would provide greater insights.

Future studies could also explore the use of Generative AI models, such as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), to create more synthetic data. These could generate high-fidelity, realistic synthetic data that mimic real attack scenarios.

In-depth analysis of normal data

Understanding the underlying signals and the semantic meanings of each message can improve the model's ability to detect subtle anomalies, thereby improving the overall efficacy of the IDS. However, the varying semantic rules across vehicle models complicate this process, reducing IDS generalizability.

Unsupervised or semi-supervised learning

As seen in this thesis and related works, supervised learning models are effective in detecting patterns and anomalies seen during training. However, these types of models are constrained by their training datasets and therefore have difficulties identifying novel, zero-day attacks. To be able to detect previously unseen types of attacks, future research should explore the integration of unsupervised and semi-supervised learning models. These models, which do not only rely on labeled data, have the potential to improve the detection capabilities for attacks without known patterns. Furthermore, unsupervised models, trained on normal data and capable of detecting anomalies, would be suitable for identifying Suspension attacks. Suspension attacks, which involve the removal of messages rather than the injection of malicious messages, can result in unexpected gaps or irregularities in patterns. Given the difficulty in labeling Suspension attacks, unsupervised learning approaches could identify anomalies without the need for labeled examples.

Deployment

There are different strategies for deploying the IDS, as discussed in Section 2.3.1. For example, the deployment on an ECU is constrained by the ECU's available computational power and memory, limiting the complexity of the implementation. Therefore, it is important that the IDS is lightweight and offers a balance between efficiency and accuracy, suitable for real-time applications in-vehicle systems. In this thesis, only a limited number of features were used from the CAN messages to reduce complexity, and networks with fewer layers were used to make it less resource-intensive. For further research, methods for reducing model size could be investigated, such as knowledge distillation, model pruning, or quantization.

Chapter 7
Conclusions

The aim of this thesis was to generate Controller Area Network (CAN) traffic attack data to develop an effective Intrusion Detection System (IDS) using deep learning techniques. Four types of attack mechanisms were investigated that can severely compromise the in-vehicle CAN bus system. These include Fabrication, Suspension, Masquerade, and Replay attacks. An attack generator was implemented that synthetically simulated attack files using a log file with normal driving data from a test truck. The attacks were injected in patterns derived from both analyzing the log file itself and open source data with real attacks. The analysis from the open source data helped determine the injection frequency, resulting in a ratio of injected attack messages compared to normal messages in every intrusion, similar to the real performed attacks.

The generated attack files were used when training and evaluating two deep learning models, Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). These models were chosen based on the time-series-based nature of the CAN bus messages, as LSTM performs well at handling time-series data. The CNN, on the other hand, is noted for its strong feature extraction capabilities suitable for identifying distinct patterns in the data. As input, three different variations were tested including pointwise and sequences of length 25 and 50. The pointwise input models performed well in detecting Fabrication attack mechanisms, as both Fuzzy and specifically DoS attacks were detected effectively. However, for pointwise inputs, neither CNN nor LSTM detected any Replay messages. The longer the sequence, the better the overall accuracy, as the Replay attack was also detected, and detection accuracy for Spoofing improved. In terms of accuracy, CNN showed slightly better results when using pointwise inputs, with an accuracy of 97.55%, while LSTM showed 97.52%. However, when using sequences, LSTM demonstrated slightly better results with 99.49% for size 25, and 99.84% for size 50, compared to CNN's 99.44% and 99.72% respectively. The CNN models, however, showed slightly better results in the metric False Negative Rate (FNR) for the Normal class, which is an important measure to reduce false alarms, where it showed 0% for pointwise and only 0.01% for a sequence of 50. Both models performed worse for a sequence length of 25, where LSTM had an FNR of 0.37%, and CNN had 0.36%. These results showed that LSTM performed slightly better with sequential data in terms of accuracy, particularly in longer sequences, whereas CNN more effectively managed to

minimize the false alarms.

This thesis highlights the limitations of using only synthetically generated data and supervised learning. The synthetic data might not fully represent real-world scenarios, potentially limiting the IDS's effectiveness in actual attacks. Moreover, the models are currently only trained to detect predefined types of attacks, restricting their ability to identify new, unknown attacks.

Future research could improve the IDS by implementing unsupervised and semi-supervised learning to improve detection capabilities for novel attacks. Developing more advanced simulation tools and using data from diverse driving conditions could also improve the robustness and generalizability of the IDS. Additionally, creating lightweight models suitable for onboard (ECU-based) deployment is crucial to integrate an IDS effectively into in-vehicle systems.

References

[1] M. L. Han, B. I. Kwak, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," *Vehicular Communications*, vol. 14, pp. 52–63, 2018, doi: 10.1016/j.vehcom.2018.09.004.

[2] F. Sagstetter, M. Lukasiewycz, S. Steinhorst, M. Wolf, A. Bouard, W. R. Harris, S. Jha, T. Peyrin, A. Poschmann, and S. Chakraborty, "Security challenges in automotive hardware/software architecture design," Grenoble, France, 2013, pp. 458 – 463, doi: 10.7873/DATE.2013.102.

[3] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental Security Analysis of a Modern Automobile," Oakland, CA, USA, 2010, pp. 447 – 462, doi: 10.1109/SP.2010.34.

[4] T. H. H. Aldhyani and H. Alkahtani, "Attacks to Automatous Vehicles: A Deep Learning Algorithm for Cybersecurity," *Sensors*, vol. 22, no. 1, p. 360, Jan. 2022, doi: 10.3390/s22010360.

[5] UNECE, "UN Regulation No. 155 - Cyber security and cyber security management system," 2021. [Online]. Available: https://unece.org/transport/documents/2021/03/standards/un-regulation-no-155-cyber-security-and-cyber-security

[6] B. S. Bari, K. Yelamarthi, and S. Ghafoor, "Intrusion Detection in Vehicle Controller Area Network (CAN) Bus Using Machine Learning: A Comparative Performance Study," *Sensors*, vol. 23, no. 7, p. 3610, 2023, doi: 10.3390/s23073610.

[7] T. Huang, J. Zhou, and A. Bytes, "ATG: An Attack Traffic Generation Tool for Security Testing of In-vehicle CAN Bus," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, no. 32. Association for Computing Machinery, 2018, p. 6, doi: 10.1145/3230833.3230843.

[8] R. B. GmbH, *CAN Specification Version 2.0*, 1991. [Online]. Available: http://esd.cs.ucr.edu/webres/can20.pdf

[9] International Organization for Standardization, "ISO 11898: Road vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication," Geneva, Switzerland, 1993. [Online]. Available: https://www.iso.org/standard/20380.html

[10] Robert Bosch GmbH, "CAN FD," 2012. [Online]. Available: https://www.bosch-semiconductors.com/ip-modules/can-protocols/can-fd/

[11] International Organization for Standardization, "ISO 11898-1:2015 Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling," Geneva, Switzerland, 2015. [Online]. Available: https://www.iso.org/standard/63648.html

[12] R. U. D. Refat, A. A. Elkhail, A. Hafeez, and H. Malik, "Detecting CAN Bus Intrusion by Applying Machine Learning Method to Graph Based Features," in *Intelligent Systems and Applications*, K. Arai, Ed. Cham: Springer International Publishing, 2022, pp. 730–748, doi: 10.1007/978-3-030-82199-$9_4$9.

[13] B. Lampe and W. Meng, "A survey of deep learning-based intrusion detection in automotive applications," *Expert Systems with Applications*, vol. 221, 2023, doi: 10.1016/j.eswa.2023.119771.

[14] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *Proceedings of the 25th USENIX Conference on Security Symposium*. USA: USENIX Association, 2016, p. 911–927, doi: 10.5555/3241094.3241165.

[15] M. E. Verma, R. A. Bridges, M. D. Iannacone, S. C. Hollifield, P. Moriano, S. C. Hespeler, B. Kay, and F. L. Combs, "A comprehensive guide to CAN IDS data and introduction of the ROAD dataset," *PLoS ONE*, vol. 19, no. 1, pp. 1 – 32, 2024, doi: 10.1371/journal.pone.0296879.

[16] A. Dehlaghi-Ghadim, M. H. Moghadam, A. Balador, and H. Hansson, "Anomaly detection dataset for industrial control systems," *IEEE Access*, vol. 11, pp. 107 982–107 996, 2023, doi: 10.1109/ACCESS.2023.3320928.

[17] A. Dehlaghi-Ghadim, A. Balador, M. H. Moghadam, H. Hansson, and M. Conti, "ICSSIM—a framework for building industrial control systems security testbeds," *Computers in Industry*, vol. 148, p. 103906, 2023, doi: 10.1016/j.compind.2023.103906.

[18] V. H. Le, J. den Hartog, and N. Zannone, "Security and privacy for innovative automotive applications: A survey," *Computer Communications*, vol. 132, pp. 17–41, 2018, doi: 10.1016/j.comcom.2018.09.010.

[19] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive CAN networks—Practical examples and selected short-term countermeasures," *Reliability Engineering System Safety*, vol. 96, no. 1, pp. 11–25, 2011, doi: 10.1016/j.ress.2010.06.026.

[20] S.-F. Lokman, A. T. Othman, and M.-H. Abu-Bakar, "Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review," *EURASIP Journal on Wireless Communications and Networking*, no. 184, pp. 1–17, 2019, doi: 10.1186/s13638-019-1484-3.

[21] H. M. Song, H. R. Kim, and H. K. Kim, "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)*, Kota Kinabalu, Malaysia, 2016, pp. 63–68, doi: 10.1109/ICOIN.2016.7427089.

[22] T.-N. Hoang and D. Kim, "Detecting in-vehicle intrusion via semi-supervised learning-based convolutional adversarial autoencoders," *Vehicular Communications*, vol. 38, no. 100520, 2022, doi: 10.1016/j.vehcom.2022.100520.

[23] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, no. 100198, 2020, doi: 10.1016/j.vehcom.2019.100198.

[24] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based Intrusion Detection System for In-Vehicle Network," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, Belfast, Ireland, 2018, pp. 1–6, doi: 10.1109/PST.2018.8514157.

[25] S. Khandelwal and S. Shreejith, "Real-Time Zero-Day Intrusion Detection System for Automotive Controller Area Network on FPGAs," in *2023 IEEE 34th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2023, pp. 139–146, doi: 10.1109/ASAP57973.2023.00033.

[26] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "Canet: An unsupervised intrusion detection system for high dimensional can bus data," *IEEE Access*, vol. 8, pp. 58 194–58 205, 2020, doi: 10.1109/ACCESS.2020.2982544.

[27] U. E. Larson, D. K. Nilsson, and E. Jonsson, "An approach to specification-based attack detection for in-vehicle networks," in *2008 IEEE Intelligent Vehicles Symposium*, Eindhoven, Netherlands, 2008, pp. 220–225, doi: 10.1109/IVS.2008.4621263.

[28] H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, "SAIDuCANT: Specification-Based Automotive Intrusion Detection Using Controller Area Network (CAN) Timing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1484–1494, Feb. 2020, doi: 10.1109/TVT.2019.2961344.

[29] V. Jadeja, A. L. N. Rao, A. Srivastava, S. Singh, P. Chaturvedi, and G. Bhardwaj, "Convolutional Neural Networks: A Comprehensive Review of Architectures and Application," in *2023 6th International Conference on Contemporary Computing and Informatics (IC3I)*, vol. 6, Gautam Buddha Nagar, India, 2023, pp. 460–467, doi: 10.1109/IC3I59117.2023.10397695.

[30] Y. Yu, X. Si, C. Hu, and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, July 2019, doi: 10.1162/neco_a_01199.

[31] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–80, November 1997, doi: 10.1162/neco.1997.9.8.1735.

[32] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451 – 2471, October 2000, doi: 10.1162/089976600300015015.

[33] H. M. Song and H. K. Kim, "Can Network Intrusion Datasets." [Online]. Available: http://ocslab.hksecurity.net/Datasets/car-hacking-dataset

[34] C. Neelap and H. V. Bhandari, "Attack Traffic Generation for Network-based Intrusion Detection System," M.S. Thesis, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden, 2023.

[35] M. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "LSTM-Based Intrusion Detection System for In-Vehicle Can Bus Communications," *IEEE Access*, vol. 8, pp. 185 489 – 185 502, 2020, doi: 10.1109/ACCESS.2020.3029307.

[36] M. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "An Effective In-Vehicle CAN Bus Intrusion Detection System Using CNN Deep Learning Approach," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, Taipei, Taiwan, 2020, pp. 1–6, doi: 10.1109/GLOBECOM42002.2020.9322395.

Printed by Tryckeriet i E-huset, Lund 2024

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2024-973
http://www.eit.lth.se