Intrusion Detection Systems to Secure In-Vehicle Networks

by

Linxi Zhang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer and Information Science)
in the University of Michigan-Dearborn
2023

Doctoral Committee:

Professor Di Ma, Chair
Assistant Professor Birhanu Eshete
Associate Professor Jinhua Guo
Professor Wencong Su

Linxi Zhang
linxiz@umich.edu
ORCID iD: 0000-0002-6233-5266

To my family

ACKNOWLEDGEMENTS

Completing this work would not have been possible without the assistance and support of the kind people around me, although it is not feasible to mention all of them here. The contribution and support of each and every one of you have been integral to the achievement of this journey. I am truly thankful.

First and foremost, I would like to respectfully express my profound gratitude to my advisor, Professor Di Ma, for her unwavering support and guidance throughout this journey. Her invaluable insights, constructive feedback, and inspirational conversations have played a crucial role in shaping my research. Her steadfast commitment to academic excellence and personal development has significantly impacted my growth as a researcher and as an individual.

My sincerest appreciation goes to my dissertation committee members: Professor Birhanu Eshete, Professor Jinhua Guo, and Professor Wencong Su. Their in-depth knowledge has added tremendous value to my work, improving its quality significantly. Learning from them has indeed been a privilege. I extend my heartfelt appreciation to the Chair of the Department, Professor Qiang Zhu, the Computer and Information Science Ph.D. Committee Chair, Professor Jinhua Guo, and esteemed professors - Anys Bacha, Brahim Medjahed, David Yoon, Jin Lu, Jie Shen, Shengquan Wang, and Zheng Song. Their invaluable advice and guidance have been fundamental in both my academic journey and personal growth.

I sincerely thank my collaborators and peers for their friendship, help, and collaboration, which have made the demanding path to a Ph.D. much more rewarding and enlightening. Particular gratitude goes to Robert Kaster and Huaxin Li for many insightful discussions and conversations. I also appreciate the strong support from the administrative and technical staff of the department. Their behind-the-scenes efforts have ensured a conducive research environment, facilitating the smooth execution of my project.

Further, I am deeply grateful to the University, the College of Engineering and Computer Science, and the Department of Computer and Information Science for their immense support in terms of funding and resources, which have been instrumental in facilitating and supporting this research.

On a personal level, I wish to express my deepest gratitude to my husband for his constant support and patience. He went through all the ups and downs with me, and without his unwavering love and support, the completion of this work would not have been possible. I am deeply grateful to my parents for their unconditional support and belief in me. They have given me endless love and encouragement over the years. My gratitude for them is beyond words. I would also like to thank my father-in-law and mother-in-law for their constant encouragement and invaluable assistance in caring for my son. And to my precious son, a special thanks for being the joy in my life.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF FIGURES ix

LIST OF TABLES x

ABSTRACT xi

CHAPTER

1 Introduction 1

1.1 Motivation and Overview 1
1.2 Challenges 3
1.3 Contributions 4
1.3.1 An Empirical Comparative Study on IDS for CAN 5
1.3.2 A Hybrid Approach Toward Efficient and Accurate Intrusion Detection for In-Vehicle Networks 5
1.3.3 Accelerating In-Vehicle Network Intrusion Detection System Using Binarized Neural Network 6
1.3.4 Efficient and Effective In-Vehicle Intrusion Detection System Using Binarized Convolutional Neural Network 7
1.4 Organization 8

2 Background 10

2.1 Controller Area Network (CAN) 11
2.2 CAN Risks 13
2.2.1 CAN Vulnerabilities 13
2.2.2 In-Vehicle Attacks 14
2.3 CAN Defenses 17

2.3.1 Message Authentication and Sender Identification . . . 17
2.3.2 Intrusion Detection Systems . . . 18
2.4 Accelerating Neural Network . . . 19
2.5 Binarized Neural Network . . . 20
2.6 Deep Learning on FPGAs . . . 21

3 Literature Review . . . 23

3.1 Rule-based IDS . . . 23
3.2 Machine Learning-based IDS . . . 24
3.2.1 General Machine Learning-based Approach . . . 26
3.2.2 Neural Network-based Approach . . . 27
3.2.2.1 Intrusion Detection Using Sequential Patterns . . . 27
3.2.2.2 Intrusion Detection Using Spatial Features . . . 27

4 Empirical Comparative Study . . . 29

4.1 Adversary Model . . . 29
4.1.1 Attack Types . . . 30
4.1.2 Adversary Types . . . 32
4.2 Data Collection . . . 33
4.2.1 Equipment and Routes . . . 34
4.2.2 Datasets and Message Characterization . . . 34
4.3 Evaluation Metrics . . . 37
4.3.1 Effectiveness . . . 38
4.3.2 Computation Time . . . 40
4.4 Rule-based IDS Comparative Study . . . 41
4.4.1 Summary of Rule-based IDS . . . 41
4.4.2 Experimental Results . . . 45
4.5 Machine Learning-based IDS Comparative Study . . . 48
4.5.1 Summary of Machine Learning-based IDS . . . 48
4.5.2 Experimental Results . . . 52

5 A Hybrid Approach Toward Efficient and Accurate Intrusion Detection for In-Vehicle Networks . . . 54

5.1 Introduction . . . 54
5.2 Our Design . . . 56
5.2.1 Stage 1: Rule-based System Design . . . 57
5.2.2 Stage 2: Deep Neural Network-based System Design . . . 59

5.3 Implementation and Evaluation . . . 63
5.3.1 Experimental Datasets . . . 64
5.3.2 Attack Strategy . . . 65
5.3.2.1 Training Datasets . . . 66
5.3.2.2 Testing Datasets . . . 67
5.3.3 Experimental Results . . . 67
5.4 Discussion . . . 75
5.5 Conclusion . . . 77

6 Accelerating In-Vehicle Network Intrusion Detection System Using Binarized Neural Network . . . 78

6.1 Introduction . . . 78
6.2 Our Design . . . 80
6.2.1 Input Generator . . . 81
6.2.2 Binarized Neural Network . . . 84
6.2.2.1 Binarization in BNN . . . 85
6.2.2.2 BNN Model Design . . . 86
6.2.2.3 BNN Inference on FPGAs . . . 87
6.2.3 Energy Consumption and Memory Cost . . . 88
6.3 Implementation and Evaluation . . . 89
6.3.1 Experimental Datasets . . . 89
6.3.2 Attack Strategy . . . 90
6.3.3 Experimental Setup . . . 91
6.3.4 Input Generator . . . 92
6.3.5 Overfitting Prevention . . . 93
6.3.6 Experimental Results . . . 94
6.3.6.1 Experiment 1: BNN vs. Full-Precision NN on CPU . . . 94
6.3.6.2 Experiment 2: BNN Implementation on FPGA, CPU, and GPU . 96
6.3.6.3 Experiment 3: BNN Models with Wider/Deeper Structure . . . 97
6.4 Discussion . . . 99
6.5 Conclusion . . . 100

7 Efficient and Effective In-Vehicle Intrusion Detection System using Binarized Convolutional Neural Network . . . 102

7.1 Introduction . . . 102
7.2 Our Design . . . 104
7.2.1 Input Generator for BCNN-based IDS . . . 105

7.2.1.1 Limitations of Prior Work . . . 106
7.2.1.2 Our Input Generator . . . 107
7.2.2 Binarized Convolutional Neural Network . . . 110
7.2.3 Accuracy Improved BCNN Model – QuickNet . . . 112
7.3 Implementation and Evaluation . . . 113
7.3.1 Experimental Datasets . . . 113
7.3.2 Attack Strategy . . . 114
7.3.3 Experimental Setup . . . 115
7.3.4 Experimental Results . . . 115
7.3.4.1 Experiment 1: BCNN-based IDS under Different Attacks . . . 115
7.3.4.2 Experiment 2: Effectiveness Evaluation on Input Generator . . . 116
7.3.4.3 Experiment 3: Effectiveness/Efficiency Evaluation . . . 117
7.3.4.4 Experiment 4: Improved BCNN Models . . . 118
7.4 Discussion . . . 119
7.5 Conclusion . . . 120
8 Conclusion and Future Work . . . 122
8.1 Conclusion . . . 122
8.2 Future Work . . . 125
BIBLIOGRAPHY . . . 127

LIST OF FIGURES

FIGURE

2.1 CAN frame format 12
2.2 CAN bus topology 12
2.3 Attack Scenario on CAN 14

4.1 Types of attacks: (a) injection attack, (b) drop attack, and (c) masquerade attack 31
4.2 Driving path of data collection 35
4.3 Distribution of time intervals for ID 0x224 37

5.1 Workflow of the proposed two-stage IDS framework 57
5.2 The DNN structure 60
5.3 A single neuron 60
5.4 Example message for spoofing attack and masquerade attack 66
5.5 Performance over Dataset 1 72
5.6 Performance over Dataset 2 73
5.7 Performance over Dataset 3 73
5.8 Performance over Dataset 4 73
5.9 Model loss 74

6.1 Workflow of the proposed BNN-based IDS 82
6.2 CAN traffic log without attack (left) and with attack (right) 82
6.3 An example input frame generated by the input generator 83
6.4 Baseline BNN structure 87
6.5 FPGA inference workflow 88

7.1 Workflow of the proposed BCNN-based IDS 105
7.2 Recurrence plots of CAN traffic using 1) only ID, without attack (a) and with attack (b); 2) ID and data, without attack (c) and with attack (d) 107
7.3 Workflow for converting CAN messages into image pixels 108
7.4 Example of converting feature vector to feature matrix 108

LIST OF TABLES

TABLE

4.1 Details of datasets 36
4.2 Confusion matrix 38
4.3 Summary of rule-based IDS 42
4.4 Experimental results for rule-based IDS 46
4.5 Summary of machine learning-based IDS 49
4.6 Experimental results for machine learning-based IDS 52

5.1 Experimental datasets 65
5.2 IDS performance - strong adversary 68
5.3 IDS performance - medium adversary 69
5.4 IDS performance - weak adversary 69
5.5 IDS performance - spares attack traffic 71
5.6 IDS performance - dense attack traffic 71

6.1 Experimental datasets 90
6.2 BNN vs. full-precision NN on CPU 95
6.3 BNN - hybrid attack implementation 97
6.4 Effectiveness evaluation - hybrid attack 98
6.5 Latency evaluation (inference time/message) 99
6.6 Model size of different models 99

7.1 Experimental datasets 114
7.2 BCNN-based IDS under different attacks 116
7.3 Input generator effectiveness 117
7.4 Effectiveness/Efficiency of BCNN scheme 118
7.5 Further improved BCNN design 119

ABSTRACT

With recent advancements in the automotive world and the introduction of autonomous vehicles, more software-driven electrical components and wireless connectivity are being introduced to increase reliability and improve safety systems for modern vehicles. However, these advanced systems also bring new risks, particularly from the security perspective, as attack surfaces are expanding and new attacks are emerging. Automotive security has become a real and important issue. For modern vehicles, in-vehicle networks that connect Electronic Control Units (ECUs) have increasingly become targets of vehicle cyberattacks. Researchers have demonstrated that attackers can intentionally control a vehicle by gaining access to the in-vehicle network, posing security and safety risks for vehicles and passengers.

Intrusion Detection Systems (IDSs) provide effective countermeasures for IT systems. However, they cannot be applied to automotive systems directly due to the challenges posed by real-time automotive systems. Some schemes for in-vehicle networks have limitations in detecting certain critical attacks, while others need to improve their efficiency to meet the challenges of real-time requirements. Existing CAN IDS designs based on *anomaly detection* usually follow two approaches: rule-based and machine learning-based. Generally, rule-based approaches are more efficient but lack abilities to detect certain types of attacks. It is difficult, if not impossible, to come up with a complete set of rules that can cover all abnormal behaviors. Machine learning-based approaches usually achieve relatively high detection accuracy but, at the same time, often involve high computational costs as well as a higher rate of false positives. Moreover, many existing schemes do not consider the real-time requirements of in-vehicle networks and the memory constraints of ECUs. In addition, their evaluations often rely on simulated data or data collected from only one or a few vehicles, given the lack of standard benchmark datasets in this domain.

This dissertation aims to address the challenges mentioned above. We (1) investigate how

existing IDS schemes work and understand their strengths and weaknesses, and conduct a detailed literate review and an empirical comparative study to provide a comprehensive understanding; (2) design a new hybrid approach for efficient and accurate intrusion detection. Specifically, we explore the combination of rule-based and machine learning-based approaches to build a two-stage IDS framework that inherits the advantages of both; (3) propose a novel IDS based on Binarized Neural Network (BNN) to accelerate intrusion detection with the consideration of real-time demand of in-vehicle networks. Furthermore, it can be further accelerated by hardware through Field-Programmable Grid Arrays (FPGAs); (4) present a new Binarized Convolutional Neural Network (BCNN)-based IDS to stick a balance between accuracy and acceleration. In particular, we design an input generator that helps machine learning models learn better for higher accuracy; (5) collect real Controller Area Network (CAN) data from seven different vehicle models from different OEMs; and (6) evaluate the proposed schemes with real CAN data. Our research includes a comparative study to provide a comprehensive understanding of existing IDSs, the development of a novel hybrid IDS framework, the exploration of BNN and BCNN to design advanced IDSs suited for in-vehicle environments, the data collection from seven different real vehicles, and the evaluation of the IDSs we proposed. Results show that our research is promising, and the proposed schemes have the potential to improve vehicle security significantly. Overall, this dissertation develops novel IDSs to effectively and efficiently protect in-vehicle networks and meet the unique challenges posed by real-time requirements of in-vehicle networks. This research provides innovative solutions to enhance the security of in-vehicle networks, ensuring vehicle security in an increasingly connected and autonomous world.

CHAPTER 1

Introduction

1.1 Motivation and Overview

The transportation industry is experiencing a revolutionary transformation led by the rapid development of autonomous and connected vehicle technologies together with powerful new mobility services. These services support new types of mobility and promise to virtually eliminate crashes and fatalities, which are chronic problems in the current landscape of the automotive world [86, 52, 80, 90]. In order to deliver these promising services, automotive manufacturers have to eliminate malicious actors in such ecosystems to minimize their impact. Automotive cybersecurity is a significant problem in today's automotive industry. With more software models and external interfaces being added to modern vehicles to support emerging mobility services, in-vehicle networks connecting tens of Electronic Control Units (ECUs) are no longer in isolated environments. They may be subject to various cyberattacks. Such attacks have been successfully demonstrated on Controller Area Network (CAN), the most predominant in-vehicle bus communication protocol, showing that the security of in-vehicle networks has become a critical issue [54, 71, 73, 72]. In these attacks, adversaries are able to access in-vehicle networks and send arbitrary CAN messages to interrupt the normal operations of the target vehicle, which may lead to serious safety consequences [73, 81, 82].

To protect in-vehicle networks from those automotive cyberattacks, two major categories of work have been proposed: 1) message authentication and sender identification, and 2) Intrusion

Detection System (IDS). Message authentication and sender identification schemes can be efficient for Internet security and provide a certain level of security but generally have a high communication overhead. Implementing authentication mechanisms for in-vehicle networks presents significant challenges because real-time communication and processing are prerequisites for these systems. Although these strategies can work on advanced ECUs or gateways with higher computational capabilities, their implementation will need altering or reprogramming every device, leading to substantial deployment challenges. Furthermore, the message space in in-vehicle networks is limited, leaving little room for additional elements such as message authentication code (MAC). Some Original Equipment Manufacturers (OEMs) are considering the use of authentication within the Controller Area Network Flexible Data-Rate (CAN-FD), which provides up to a 64-byte payload. However, this solution is primarily designed for new vehicles, and it is generally not feasible to retrofit it onto existing vehicles already on the roads. Importantly, it should be highlighted that a successfully implemented authentication scheme can become ineffective if one or more ECUs become compromised. Because these ECUs are a legitimate part of the vehicle and are often configured with the correct credentials for secure communication, a breach in their security can render the authentication method ineffective. For IDSs, they generally cost no additional communication overhead, and their implementation involves integration as extensions on a single ECU or a gateway, greatly simplifying the installation process. Typically, an IDS protects in-vehicle networks by continuously monitoring network traffic and identifying potentially malicious messages.

In-vehicle networks enable ECUs to coordinate critical functions like transmission, engine, body, and chassis control. Unexplored vulnerabilities and unresolved issues can cause security or safety issues for both vehicles and passengers. In this work, we aim to protect vehicles by developing efficient IDS systems that consider the real-time reliability demands of in-vehicle networks.

1.2 Challenges

Efforts from various sectors - academia, industry, and government bodies - have been dedicated to enhancing vehicle security. The need to secure in-vehicle networks is critical as it directly relates to driver and passenger safety. We must enhance the security of in-vehicle networks, which is the core of vehicles. Anomaly-based IDSs are widely recognized as an effective strategy to secure these networks and thwart malicious attacks. It is essential to investigate 1) whether existing IDS schemes work effectively and efficiently, and 2) whether IDS schemes meet the demand of real-time communications of in-vehicle networks.

Existing IDS Schemes for In-Vehicle Networks: Currently, there are two primary approaches used for intrusion detection: rule-based and machine learning-based. For rule-based approaches, various "rules" based on characteristics (such as periodicity, frequency, or entropy) of normal CAN messages are derived, and CAN messages not following these rules are considered potentially malicious [71, 14, 101, 76, 69, 122]. Usually, these systems are computationally inexpensive. However, it is a challenging task, if not impossible, to construct a complete set of rules that can cover all normal behaviors of the CAN traffic, especially in complicated situations where messages may be sent non-periodically or on demand. Consequently, in realistic attack scenarios, a rule-based system can provide low detection rates. In order to achieve higher detection rates, various machine learning techniques, such as deep neural networks, are introduced to design new generations of IDSs [125, 78, 49, 50, 126, 57, 102, 97, 130]. However, machine learning-based IDSs can suffer from high false positives and typically involve high processing latency and computational costs.

Real-Time Communication Demands in In-Vehicle Networks: Advanced machine learning algorithms have been utilized for intrusion detection. Neural networks have shown remarkable capabilities in complicated applications, including image classification, semantic segmentation, and object detection [59, 45]. However, they generally have high computational complexity, usually require considerable computing resources, and often result in large power consumption. A neural network model can require hundreds of Megabytes (MBs) of memory [100, 140, 6, 39]. This requirement for memory access is often the bottleneck of system performance as well as

energy efficiency [104]. Deploying complex neural networks on resource-constrained devices in embedded environments can be challenging. Given the real-time communication requirements of the in-vehicle networks, an IDS for such networks needs to be compact and exhibits fast detection response. Another crucial consideration is the inherent trade-off between accuracy and speed. It is a widely accepted principle that, in general, in order to achieve faster processing speed, a degree of accuracy can be sacrificed. Therefore, the challenge is to balance efficient and accurate IDS to ensure reliable and secure in-vehicle networks.

1.3 Contributions

Improving the security of in-vehicle networks, and by extension, the vehicles themselves, is critical as it directly impacts the safety of drivers and passengers. In order to enhance the security of in-vehicle networks, it is important to evaluate existing state-of-the-art defense schemes. We need to analyze whether existing schemes fail to handle threats effectively and efficiently. Also, we must consider whether IDS schemes are suitable for fulfilling the real-time communication demands in in-vehicle networks. Through such an understanding, we design new IDSs to secure in-vehicle networks better and further ensure the safety of vehicles.

In this dissertation, the main contributions are as follows: (i) We perform a comparative study of various IDSs. Representative existing schemes are implemented and evaluated on the same platform based on the same real CAN data, adversary model, and evaluation metrics. The analysis reveals insights into the strengths and weaknesses of these systems, providing a comparative understanding and guidance for future advancements in CAN IDS. (ii) We design and implement a new IDS framework for efficient and accurate intrusion detection. The proposed IDS combines traditional rule-based IDS techniques with emerging machine learning methods to achieve a balance between detection accuracy and efficiency. It is capable of detecting various types of attacks, including those that many schemes cannot. This hybrid IDS framework also allows future updates, and each component can be replaced by other rules or machine learning algorithms accordingly. (iii) We

propose and implement a new IDS to accelerate in-vehicle network intrusion detection by using a Binarized Neural Network (BNN). Considering the embedded environment of in-vehicle networks and real-time demands, the proposed IDS aims to provide faster detection, smaller memory cost, and lower power consumption. The IDS is also compatible with hardware acceleration techniques for further acceleration. (iv) We develop and implement a new IDS utilizing Binarized Convolutional Neural Network (BCNN) for higher accuracy while maintaining the benefits of BNN. The proposed IDS keeps the advantages of binarization while striving for a balance between accuracy and acceleration, ultimately aiming to achieve improved accuracy.

1.3.1 An Empirical Comparative Study on IDS for CAN

We conduct an empirical comparative study of in-vehicle network IDSs in the existing literature. The focus of this study is to investigate their strengths and weaknesses. Each IDS is implemented and evaluated on the same platform using the same dataset, adversary model, and evaluation metrics, which ensures a fair and comprehensive comparison. This study aims to provide a comprehensive understanding and valuable insights, guide future research directions, and help design robust and efficient in-vehicle network IDSs, ultimately helping to advance the field of automotive cybersecurity.

1.3.2 A Hybrid Approach Toward Efficient and Accurate Intrusion Detection for In-Vehicle Networks

Currently, there are two primary approaches used for intrusion detection: rule-based and machine learning-based. Our analysis shows that existing rule-based or machine learning-based IDSs have their own limitations. Rule-based approaches are efficient but can have low accuracy due to limitations in detecting certain types of attacks which are often sophisticated and common, and can occur in real attack scenarios. Machine learning-based approaches have comparably higher detection accuracy but, at the same time, have high computational costs. Moreover, machine

learning-based approaches also have inherent limitations in detecting certain types of attacks. To overcome those limitations and defend against various vehicle attacks, we propose a new hybrid IDS framework with two stages. The proposed scheme combines traditional rule-based IDS techniques with emerging machine learning methods to achieve a balance between detection accuracy and efficiency. More specifically, we use machine learning methods to achieve a high detection rate while keeping the low computational requirement by offsetting the detection with a rule-based component. Our experiments with CAN data collected from four different real vehicle models demonstrate the effectiveness and efficiency of the proposed hybrid IDS. An additional benefit of this IDS is its flexibility as a framework, where the rule-based or machine learning components can be replaced based on specific needs or requirements. This offers the advantage of adapting the IDS to different scenarios and optimizing its detection capabilities on a case-by-case basis. Our experimental evaluations on four real vehicle datasets demonstrate that the proposed IDS detects various types of attacks with a low false positive rate of 0.066%.

1.3.3 Accelerating In-Vehicle Network Intrusion Detection System Using Binarized Neural Network

Machine learning-based IDSs usually exhibit higher detection accuracy. However, those systems generally involve high latency, require considerable memory space, and often result in high energy consumption. To accelerate intrusion detection and also reduce memory and energy costs, we propose a new IDS system using BNN. Compared to full-precision counterparts, BNNs offer faster detection, smaller memory cost, and lower energy consumption. Moreover, BNNs can be further accelerated by leveraging Field-Programmable Grid Arrays (FPGAs) since BNNs cut down hardware consumption. The proposed IDS is based on a BNN model that suits CAN messages and takes advantage of sequential features of messages rather than each individual message. We also explore various design choices for BNN, including increasing network width and depth, to improve accuracy since BNNs typically sacrifice accuracy. The performance of our IDS is evaluated with four different real vehicle datasets. Experimental results show that the proposed IDS reduces

the detection latency (3 times faster) on the same CPU platform while maintaining acceptable detection rates compared with full-precision models. Furthermore, we examine the proposed IDS on multiple platforms, and our results show that using FPGA hardware reduces the detection latency dramatically (128 times faster) with lower power consumption compared to an embedded CPU device. In addition, we evaluate BNNs with different designs. Results demonstrate that wider or deeper models definitely improve accuracy at the cost of increased latency and model sizes to varying degrees. Applications are recommended to choose the appropriate model design for their needs based on the available resources they have.

1.3.4 Efficient and Effective In-Vehicle Intrusion Detection System Using Binarized Convolutional Neural Network

Current BNN-based IDSs, while providing acceleration benefits, often fall short when it comes to offering high accuracy in securing CAN. Our goal is to maintain the advantages of BNNs while improving their accuracy. To this end, we propose a novel IDS utilizing BCNN. This IDS fits CAN traffic and takes advantage of temporal and spatial features of CAN messages instead of individual messages. In particular, we design an input generator that exploits the temporal sequential pattern and spatially local correlation of messages to facilitate model learning and ensure high-accuracy performance. Our experimental results show that the proposed IDS effectively reduces memory cost and detection latency while maintaining high detection rates. Specifically, our IDS runs 4 times faster and utilizes only 3.3% of the memory space required by a full-precision CNN-based IDS. At the same time, it achieves 90.2% of the accuracy of the CNN-based IDS and improves 11.9% of the accuracy of the state-of-the-art BNN-based IDS design.

Scope of This Dissertation

Focus on CAN: In-vehicle networks include Local Interconnect Network (LIN), CAN, FlexRay, CAN with Flexible Data-Rate (CAN-FD), Media Oriented Systems Transport (MOST), Ethernet, etc. In this work, we primarily focus on CAN among those protocols. As the *de facto* standard

for in-vehicle networks, CAN is relatively low-cost, efficient, and reliable. CAN is mandatory for all cars sold in the US, and almost all new passenger cars manufactured in Europe are equipped with it. In addition to automotive usage, CAN is adopted by other industries, including railways, aircraft, aerospace, and medical equipment. Since these applications are directly related to safety concerns, it is crucial to secure CAN effectively.

Focus on Anomaly Detection: There are two intrusion detection methods: anomaly-based and signature-based. We delve into anomaly-based IDS to protect in-vehicle networks against malicious activities. The anomaly-based method utilizes a normalized baseline and monitors deviations between the current traffic and the baseline. Once an abnormal event occurs, an alert is triggered. The normalized baseline is generated by analyzing the normal traffic of the protected vehicle. The signature-based method, also called misuse detection, leverages predefined attack signatures to monitor current traffic and detect intrusions. This method accurately identifies attacks corresponding to the attack signatures, offering high precision with minimal false positives. However, it lacks the ability to catch attacks that are not predefined. Currently, knowledge of attack signatures is limited, and there is no comprehensive database for vehicle cyberattacks. Hence, this method is not as developed as anomaly-based detection for in-vehicle networks. Our focus is primarily on anomaly detection, which is able to identify any abnormal state in addition to the attack state.

When we mention "rule-based" methods, we specifically refer to a subtype of anomaly detection methods: the rule-based anomaly detection methods. These rules are created by studying the characteristics of normal in-vehicle network data. Then, they are applied to differentiate between normal and abnormal behaviors.

1.4 Organization

This dissertation is organized as follows. Chapter 2 introduces the detailed background, including an overview of in-vehicle networks, existing risks, and defense countermeasures against in-vehicle

attacks. Chapter 3 provides a literature review on IDS research for in-vehicle networks. Chapter 4 presents a comparative study of existing IDSs. Chapter 5 proposes a novel hybrid IDS framework designed for in-vehicle networks. Chapter 6 introduces a binarized neural network approach to accelerate in-vehicle network intrusion detection. Chapter 7 demonstrates an IDS utilizing binarized convolutional neural network to improve detection accuracy while maintaining an accelerated detection process. Finally, we conclude the dissertation and discuss future work in Chapter 8.

CHAPTER 2

Background

With recent advancements in the automotive industry and the emergence of autonomous and connected vehicles, the security of the Controller Area Network (CAN) has become increasingly crucial. Modern vehicles are no longer isolated mechanical systems, and electronic controls and x-by-wire systems now control almost every aspect of a car. Furthermore, the evolving reality that cars will soon be driving autonomously, as well as increasing connectivity between vehicles and other elements (V2X), further highlights the importance of automotive cybersecurity.

To evaluate the capability of attackers, researchers demonstrate that attackers are able to access the CAN network via a variety of attacking interfaces such as tire-pressure monitoring system (TPMS), Bluetooth, telematics, and on-board diagnostics II (OBD-II) and take the whole control of the victim vehicle [13, 73, 81, 82, 51, 21]. The authors of [73] present how to control a Jeep Cherokee which is driving on a highway by manipulating the compromised ECU remotely. The researchers design and launch several remote attacks on the Tesla Model S/X in both Parking and Driving mode [81, 82]. Their studies show that attackers are able to access the Autopilot electronic control unit (ECU) in the vehicle and execute remote attacks by exploiting vulnerabilities.

In this chapter, we first provide an overview of in-vehicle networks, and then we introduce the risks of CAN and discuss attack procedures. Next, we review related defense countermeasures, including message authentication and sender identification, and intrusion detection systems. Finally, we provide an overview of various techniques related to effective IDS design.

2.1 Controller Area Network (CAN)

In this work, we focus on CAN, the most widely used protocol for in-vehicle communication. In addition to CAN, there are multiple protocols for in-vehicle data communications, such as FlexRay and Local Interconnect Network (LIN). FlexRay is designed to provide fast and reliable communication, but it has more complicated operations and is more expensive than CAN [4, 88]. LIN is usually used in low-speed applications for non-critical functions. CAN is the most widely used in-vehicle communication protocol for real-time and safety-related components. It is able to build reliable interconnections between ECUs/nodes through a multi-master broadcast serial bus system for safety-critical functions, including the powertrain, engine management, transmission system, and anti-brake system [107, 20, 16].

CAN Frame: The CAN bus is a message broadcast bus transmitting messages that carry up to 64-bit data. In the bus, data is exchanged among ECUs through messages which help maintain data consistency and deliver information among ECUs for making control decisions. Figure 2.1 illustrates the structure of a CAN frame, a universal format adhered to by all CAN messages for data transmission. A message includes: Start-of-Frame (1 bit), Identifier (11 bits or 29 bits), Control field (6 bits), Data field (0-8 bytes), CRC field (16 bits), ACK field (2 bits), End-of-Frame (7 bits) and Inter-Frame Space (3 bits). Note that a frame featuring an 11-bit ID is a standard CAN frame, aligning with the CAN 2.0A protocol, while a frame carrying a 29-bit ID suggests an extended CAN frame, supporting the CAN 2.0B protocol. The latter protocol maintains compatibility with CAN 2.0A. As per the CAN standard, every CAN implementation must support the standard frame format and be resilient to the extended frame format. Furthermore, there is an extension to the original CAN, known as CAN with Flexible Data-Rate (CAN-FD), which expands the ID field to encompass up to 512 bits.

Message Broadcast: CAN bus is a message-oriented network. Each CAN message has no addressing scheme. That is, messages are not transmitted from one node to another node based on addresses. Instead, messages are assigned CAN IDs for communication. Each ECU is configured to accept CAN messages with specific IDs and disregard others at compile time. This design

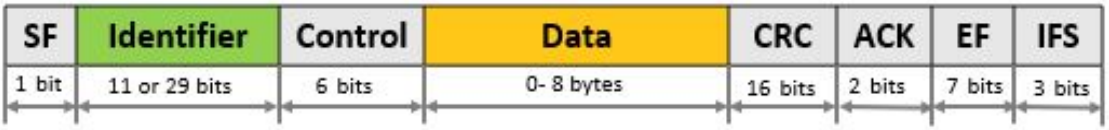

Figure 2.1: CAN frame format

offers flexibility; for instance, adding an ECU does not affect existing nodes. When a message is broadcasted, every ECU on the bus receives it. If the message's ID does not match an ECU's pre-configured list, it will be discarded; otherwise, it will be accepted and processed further. Figure 2.2 shows a typical CAN bus data transfer process. When ECU 1 sends out a message, both ECU 2 and ECU 3 receive it. ECU 2, after checking the message's ID and finding it does not match its specifications, discards the message. On the other hand, ECU 3 recognizes the ID and, as it is configured to accept such messages, processes it further.

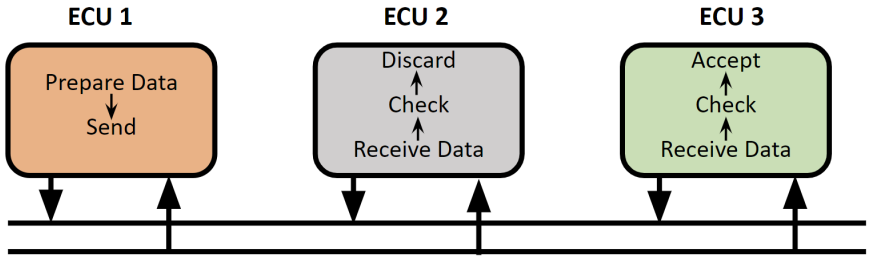

Figure 2.2: CAN bus topology

CAN Bus Arbitration: CAN operates as a broadcast communication network, leveraging message priority for collision detection and arbitration. Each ECU first evaluates the bus status before transmitting a message, ensuring to proceed only if the bus is idle. If the bus is occupied, the ECU will postpone its transmission to the next available opportunity. When several ECUs try to transmit messages at the same time, conflicts can occur. However, the CAN arbitration mechanism addresses and resolves these issues. CAN arbitration operates on the principle that a message's priority is inversely proportional to its ID value – a lower ID corresponds to a higher message priority [20, 18, 107, 16]. Imagine a situation where three CAN messages, labeled with IDs

0x0002, 0x003, and 0x00AF, are being sent to the CAN bus at the same time. Based on the arbitration rules, the message with ID 0x0002 will be prioritized and broadcasted first because it has the highest priority given its lowest ID value. The other messages with higher ID values, indicating lower priorities, will wait their turn for subsequent transmission.

Typical CAN Setup: In a typical CAN setup, every ECU is associated with a unique set of message IDs for sending messages. A message from an ECU is assigned with an ID from the unique set of that ECU. Since message ID sets of different ECUs are unique and non-overlapped with each other, messages with the same ID must be sent from the same ECU. In other words, a valid ID corresponds to one and only one sender ECU. Each ECU is also configured to accept messages based on another subset of message IDs, and those subsets might overlap across different ECUs. This setup allows a single message sent by one ECU can be accepted and processed by multiple ECUs.

2.2 CAN Risks

2.2.1 CAN Vulnerabilities

CAN is developed to provide reliable communications with high efficiency and low cost. Although CAN physical layer has strong error detection through CRC, bit stuffing, etc., it has no security protection. The CAN vulnerabilities come from the facts: it is a broadcast bus; each node decides if it should process the message; message priority is based on ID value; messages are not encrypted or authenticated. Such design results in many potential vulnerabilities for CAN: 1) all nodes see the entire traffic as plain text, allowing eavesdropping and learning the patterns of the target ECU; 2) any (including the outside intruder and the inside intruder - a compromised ECU) node can send any arbitrary message and no one knows who send that message; 3) the broadcast nature and relying on arbitration to win bus access makes DoS attack possible, and 4) answers to standard challenges that are needed for authentication especially when doing sensitive things, such as reflashing components or firmware update, are stored in memory, etc. The aforementioned vulnerabilities make in-vehicle

networks vulnerable to attacks. For example, since there is no sender ECU identification, the attacker who compromises an ECU can send malicious messages that disrupt the bus and even cause control system failure without leaving a trace.

2.2.2 In-Vehicle Attacks

In order to enhance the security of in-vehicle networks, it is crucial to study various types of attacks, as researchers have demonstrated various techniques for launching attacks. In this section, we introduce a general attack procedure for in-vehicle networks, review existing interfaces for automotive cyberattacks, and summarize attack methodologies used in related studies.

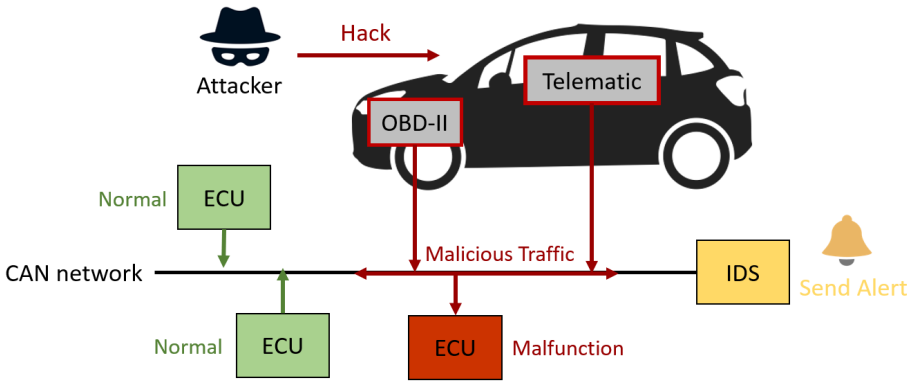

Figure 2.3: Attack Scenario on CAN

General Attack Procedure: Figure 2.3 shows a general attack scenario for CAN. A general attack procedure can be divided into three phases [66, 94]: investigation phase, preparatory phase, and attack phase. In the investigation phase, to access the target in-vehicle network, attackers determine an interface that can be used, such as the OBD-II port and telematics system. Malicious nodes can then be built through the chosen interface to launch an attack. The malicious node can be a laptop, an external ECU, a compromised ECU, or a telematics system infected by malware. In the preparatory phase, the malicious node eavesdrops and analyzes CAN messages transmitted on the bus. Since messages are broadcast to all the ECUs deployed on the CAN bus, all transmitted CAN messages can be sniffed and recorded. Attackers are able to further analyze the historical

CAN trace and build malicious CAN messages to launch attacks. In the attack phase, attackers are able to implement various types of attacks, such as injecting malicious messages. It is noted that attackers have to repeat the preparatory phase when they have new target vehicles because the implementation of CAN is different from vehicle models. Therefore, malicious messages cannot be reused to attack other different vehicle models.

Available Interfaces: Multiple interfaces can be leveraged by adversaries to access in-vehicle networks of a modern vehicle, such as the OBD-II port, CD player, USB port, and telematics systems such as GM's OnStar, Audi's Connect and BMW's Connected Drive [13, 66]. The OBD-II port supports on-board diagnostic standards, and it is designed for diagnosing vehicles and reprogramming the firmware of ECUs. It not only offers diagnostic and emissions measurement information but also additional information, such as body control, engine control, chassis control, etc. Most experimental attacks for in-vehicle networks are implemented through the OBD-II port. A laptop connecting the OBD-II port can monitor and record the messages transmitted on the CAN bus. Besides, some commercial OBD-II scan tools provide wireless connection options for smartphones, which makes wireless attacks possible such as through Bluetooth or Wi-Fi. Another interface is the telematics system that supports media entertainment, the Global Position System (GPS), and the cellular network. Modern vehicles are commonly equipped with a multi-functional telematics system to connect to external networks, which makes the telematics system become a target of vehicle cyberattacks. By exploiting available interfaces, adversaries are capable of accessing a target in-vehicle network. They can then launch various types of attacks and even intentionally control the target vehicle.

Attacking Methodologies: Several methodologies have been proven to be effective in existing studies [13, 66], and they are summarized as follows:

- Message Sniffing: Because of the broadcast nature of CAN, a malicious node can eavesdrop on all the messages transmitted on the CAN bus. By accessing an in-vehicle network via available interfaces such as the OBD-II port and telematics system, adversaries are able to monitor the real-time traffic on the bus and record all transmitted messages. Adversaries can

know details of the CAN messages by analyzing recorded messages. As the range of valid CAN messages is not large, some functions of certain ECUs can be revealed by conducting fuzzing tests [54].

- Message Injection: Adversaries can send malicious messages to the CAN bus by controlling a malicious node. The malicious node can be a laptop connecting the OBD-II port, an external ECU connecting to the bus, or a reprogrammed/compromised ECU. Because of the broadcast nature of CAN, malicious messages can be transmitted to the bus and listened to by all nodes. To attack a certain ECU, attackers can inject a carefully crafted malicious message with an appropriate ID accepted by the target ECU.

- Message Falsifying: Adversaries can design malicious messages to launch specific attacks based on an analysis of the semantic meaning of CAN messages. Those malicious messages contain altered data that can mislead or even control corresponding legitimate ECUs. For example, adversaries are able to launch attacks by falsifying messages to modify the reading of the speedometer, the fuel level, the information shown on the instrument panel cluster, or even affect the function of the transmission system. Such attacks can not only fool the driver but also cause critical safety issues.

- Message Replay: When adversaries have some valid messages, they can send such messages back to the CAN bus via a malicious node. ECUs are not able to check the freshness of the messages as well as the source of the received messages, because the CAN protocol has no authentication mechanism. This type of attack can cause critical safety issues without much effort for adversaries as it can be easily implemented. For example, an attacker is able to operate a stationary car by injecting valid messages related to the engine control into the in-vehicle network.

- Denial-of-Service (DoS) Attack: As the ID of each CAN message indicates the priority, adversaries can leverage it to launch DoS attacks. During this attack, messages with the highest priority are injected into the CAN bus, and normal nodes have to wait for the next

ideal status of the bus. This attack can be easily implemented by injecting CAN messages with the highest priority all the time. It can disable communications of normal ECUs on the bus and result in critical safety issues.

2.3 CAN Defenses

To protect CAN from attacks, various schemes are proposed for CAN defenses. Two main lines are 1) message authentication and sender identification, and 2) intrusion detection system.

2.3.1 Message Authentication and Sender Identification

Various message authentication or sender identification schemes have been proposed for CAN to defend against cyberattacks. Several cryptographic protocol proposals, most of them based on the use of message authentication code (MAC), have been proposed for **CAN message authentication** [83, 106, 105, 96, 118, 63, 85, 27, 28, 115]. However, due to the highly restricting space in CAN message (a CAN message is at most 8 bytes in length) and the demanding real-time requirement, to have a practical and deployable solution for CAN authentication is still a challenging job.

To improve efficiency for real-time detection, an anonymous ID scheme is proposed to provide *implicit* **sender identification** to prevent broadcasting from unauthorized senders in [32]. Since both the sender and target receiver can generate the anonymous IDs beforehand, this scheme is efficient and only adds negligible delay to the identification process. An *explicit* **sender identification** scheme is proposed based on the fingerprinting of ECUs by using clock skews [14].

All the message authentication and sender identification schemes aim to protect the integrity and verify the originality of CAN messages. Even though these schemes can be effective in defending against attacks originating from unauthorized devices which have access to the CAN, they are limited to defending against attacks originating from the inside - compromised ECUs. Because these ECUs are legitimate parts of the vehicle, they are usually configured with the right credential to conduct secure communication.

2.3.2 Intrusion Detection Systems

Parallel to authentication methods, intrusion detection system (IDS) schemes are designed to secure in-vehicle networks. IDSs are able to detect a wide variety of threats in real time and can be updated as new threats emerge. Moreover, an IDS can easily be integrated into existing vehicles, a convenience not often found with other methods like message encryption or authentication. The ability to retrofit an ECU or a gateway equipped with an IDS, without necessitating changes to the existing vehicle architecture, highlights the practicality and versatility of IDSs. These IDSs are usually rule-based or machine learning-based, working to detect malicious activities by distinguishing between normal and abnormal behaviors.

Rule-based IDSs leverage the features derived from static analysis of CAN messages, such as their consistent intervals [101, 69]. However, this approach encounters limitations when dealing with aperiodic or inconsistent messages. Other detection features can be the entropy of CAN messages, the fluctuation in system entropy, the message sequence, or the inter-message time distribution. However, IDSs, relying on sequence pattern recognition, are susceptible to replay attacks, resulting in high false alarm rates and low detection rates. These rule-based IDSs may find it challenging to cope with complex types of attacks that are not defined within their threat models, resulting in lower detection rates.

Given the rising complexity of attacks, advanced data analysis techniques like machine learning and neural network are being used to improve detection [50, 94, 58]. Algorithms such as deep neural network (DNN) have been employed to build anomaly detection models. Even so, machine learning-based IDSs, while typically offering superior accuracy, often fail to consider the real-time requirements and resource constraints of the CAN bus environment. Furthermore, they may be inadequate in detecting certain kinds of attacks, and some systems need to preprocess CAN data, causing processing delays and providing limited information for intrusion detection. In addtion, processing time evaluations are only reported by a few IDSs studies, and detection time can significantly vary based on system complexity, emphasizing the need for further improvement and consideration of real-time requirements of in-vehicle networks.

With the continuous evolution of automotive technologies, maintaining effective and robust security measures is critical. IDSs, given their adaptability, potential real-time detection capabilities, and ease of integration into existing vehicles, can help to address CAN vulnerabilities and ensure a secure in-vehicle network environment. Existing IDSs may perform effectively against specific threats but struggle when faced with more sophisticated attacks not covered in their threat models. There is an increasing necessity to incorporate considerations of real-time performance in the development of future IDSs for in-vehicle networks.

2.4 Accelerating Neural Network

The design and implementation of efficient IDS for in-vehicle networks need techniques that maximize performance while minimizing resource usage. Given that vehicles typically have limited computational resources, it is crucial to optimize the computational efficiency of neural network-based IDS models. Several techniques have been proposed to address this challenge, such as pruning, quantization, and hardware acceleration.

Model pruning is a strategy for neural network optimization that involves eliminating unimportant or redundant parameters from the network. It is a technique that is designed to reduce the computational complexity and memory footprint of the model without significantly affecting its performance. Early work by Han et al. [34] demonstrates the effectiveness of pruning neural networks for improved efficiency. Their method removes connections with small weights, reducing the model size without significantly impacting accuracy. More recent research has explored structured pruning, where entire neurons, layers, or feature maps are removed, enabling more significant computational savings [60, 40].

Quantization is another popular method for accelerating neural networks. It reduces the precision of the weights and sometimes the activations in the network, reducing memory usage and computation requirements. Several works have shown that neural networks can maintain reasonable performance with lower-precision weights, often as low as 1-2 bits [95, 43]. Quantization not only

reduces the memory footprint but also allows for the use of lower-precision arithmetic, leading to faster computation [17].

Hardware acceleration is a direct approach to improve the computation speed of neural networks. Specialized hardware accelerators like Graphics Processing Units (GPUs), Tensor Processing Units (TPUs), and Field-Programmable Gate Arrays (FPGAs) are used to expedite the calculations in neural networks. For instance, GPUs have long been used to accelerate neural network computation due to their parallel processing capabilities [55]. More recently, custom accelerators like TPUs have been developed specifically for accelerating neural networks [48]. FPGAs have also been used due to their reprogrammable nature and efficient parallelism for certain types of computation [119].

2.5 Binarized Neural Network

Due to the high computation cost and the large model size, advanced neural network techniques can be challenging to deploy in embedded environments. Some studies aim to make deep learning faster and smaller without sacrificing overmuch accuracy. Han et al. [33] discuss that neurons not contributing much to the network can be removed from the network. The pruning network is sparser and potentially smaller with fewer calculations. Iandola et al.[46] present a method to reduce the number of parameters for network size reduction. Both aforementioned methods can make the model size smaller to some extent, but they cannot guarantee a big size reduction when most neurons in the network significantly contribute to the network or the model is not over-parametered. Researchers propose a quantized network in the extreme case: Binarized Neural Network [17, 8]. They introduce a method to train neural networks with binary weights and activations. The result of BNNs is obvious compared to 32-bit DNNs [17]. BNNs can require up to 32 times smaller memory size and 32 times fewer memory accesses. However, BNNs may not be able to provide higher accuracy compared to their corresponding full-precision models. Various methods have been reported to improve BNN accuracy [12]. Some studies focus on improving a model structure. Bulat et al. [12] design a hierarchical network and suggest making each layer wider by adding more

neurons in hidden layers. Tang et al. [108] propose to use partial binarization during training, and only binarizing groups of kernels that have a greater impact on overall performance. Another direction of improving the accuracy of BNNs is to extend core principles of binarization itself, such as scaling with a gain term. Gain terms [95] can be used to give more capacity to a network when multiple gain terms are used within a dot product or to form a linear combination of parallel dot products.

2.6 Deep Learning on FPGAs

Deep learning applications with demanding real-time responses rely on parallelism computing in inference phases [31]. In the early years, since GPUs are specifically designed for video and image rendering, using GPUs for deep learning became widely accepted. GPUs are able to process numerous arithmetic operations in parallel so that they can offer considerable acceleration. However, compared with Application-Specific Integrated Circuits (ASICs), which are specially optimized for deep learning applications, GPUs do not deliver as much energy efficiency as ASICs at the same performance. While there is no single hardware architecture working perfectly for all deep learning applications, FPGAs provide distinct advantages over GPUs and ASICs in certain use cases. FPGAs offer flexibility and cost efficiency with circuitry that can be reprogrammed for desired functionalities. In comparison to GPUs, FPGAs provide decent performance in deep learning applications in which low latency and high power efficiency are critical. Compared with ASICs, FPGAs can be fine-tuned even after being manufactured to set a balance of power efficiency and performance with specific requirements. As accelerator devices, FPGAs can be used in the deep training phase, which has already been widely deployed in the cloud server by leading technology companies, such as Microsoft Azure [92]. It is also feasible to use FPGAs as embedded devices in the inference phase. Unlike powerful and expensive FPGAs in cloud servers, embedded FPGAs have limited logic resources and memory bandwidth, which makes them difficult to perform a full-size, full-precision deep learning inference [141]. To bridge this gap, one of the approaches

[135, 30] is to optimize deep learning computation and memory access in FPGAs by efficiently increasing the utilization of the design space. Another approach proposed in [62] is to simplify the deep learning model by reducing the precision of floating-point operations and then fit this size-reduced model onto an embedded FPGA device.

CHAPTER 3

Literature Review

The growing complexity of vehicle systems, the increasing connections between vehicles and the external environment, along with emerging cyber threats, highlights the urgent need for effective security measures. Intrusion Detection Systems (IDSs) offer a crucial line of defense against these threats. In this chapter, we explore the literature on IDS and emerging trends, and delve into details of effective IDS design and related techniques.

3.1 Rule-based IDS

Various rule-based IDSs, which exploit different characteristics of CAN messages, have been proposed over the years [128, 94]. Certain proposals leverage the fact that most CAN messages are sent at fixed intervals. For instance, the study by Song et al. detects message injection attacks by analyzing traffic anomalies based on an assumption that all CAN messages are generated at a regular frequency or interval [101]. The methods proposed in [7] rely on analyzing distributions in inter-message time and monitoring changes during the detection. Similarly, the clock-based IDS proposed by Cho et al. uses the periodic nature of many CAN messages to detect anomalies and fingerprint ECUs [14]. While lightweight and efficient, these time-interval approaches fall short when faced with attacks involving aperiodic messages.

Other approaches focus on monitoring the entropy of CAN messages or changes in system entropy for intrusion detection. The methods proposed by Muter et al. and Marchetti et al. rely on analyzing distributions in inter-message time and tracking entropy changes during the

detection process [76, 69]. Furthermore, the sequence pattern of CAN message ID can also be leveraged for intrusion detection [68]. Nevertheless, sequence pattern-based IDSs often struggle in defending against replay attacks, as the message sequences of such attacks have been observed during the modeling process and are therefore marked as genuine. Moreover, these systems, which only exploit recurring patterns of two consecutive messages, are ineffective against well-designed message injection attacks. As a result, they yield a low detection rate and high false alarm rate, presenting a critical issue for IDSs.

In addition, various other information has been explored for intrusion detection systems. For example, an IDS based on the entropy of bits in the ID section is proposed by Wang et al [122]. However, their results show that only the detection of higher-priority CAN messages achieves a higher detection rate, and their conclusions are only based on one CAN trace collected from one vehicle. Given the varied CAN implementations across different vehicles from different manufacturers, their scheme may provide unreliable performance over different datasets. An IDS based on the Hamming distance is proposed by Stabili et al. [103], but it shows poor performance on replay attacks. Lastly, Muter et al. suggest using a set of different in-vehicle sensors to verify message formality, location, data range, and data plausibility, but without implementation [77].

In summary, the aforementioned schemes usually perform well with specific threat models, but they may miss sophisticated attacks not included in those models. As such, these IDSs demonstrate a need for further improvement.

3.2 Machine Learning-based IDS

Given cars have become more connected and complicated, attacks are becoming more sophisticated. Some advanced data analytic techniques such as machine learning and deep neural network (DNN) are considered to improve detection rates, especially in detecting more sophisticated attacks or unknown attacks which can escape free from being caught by detection methods exploiting regularity, periodicity, or other CAN data characteristics, such as simple sequence patterns. Ma-

chine learning-based IDSs for in-vehicle networks have been proposed for applying hidden patterns [79, 111, 49, 125, 126, 102, 97, 42, 130]. These schemes use different machine learning algorithms as well as various data features to train the model and detect anomalies. In [79] and [125], they use data extracted from the OBD-II port to train the machine learning models for detecting anomalous activities in vehicles. The data used in [125] has clear semantic meanings about the control system by using OBD-II parameter IDs. As inputs are interpreted data, it adds delay in data processing. Moreover, since only a limited set of messages can provide clear semantic meaning, the proposed IDS has a limited amount of information to rely on for anomaly detection. The work of [79] uses the Hidden Markov Model while [125] and [102] use Artificial Neural Network (ANN) and Convolutional Neural Network (CNN), respectively. In [111], a one-class support vector machine is used to detect deviations from normal frequencies of CAN messages. The bit pattern in the data field (64-bit) of CAN messages and a DNN model are used for intrusion detection [49]. Results show that the DNN-based approach outperforms ANN-based approaches in detection accuracy. Generative Adversarial Networks (GAN) based IDSs are proposed in [97, 130]. Note that [130] requires DBC (Data Base CAN) file, which is kept strictly confidential by the automotive OEM, to train their GAN model. The authors of [126] propose a vehicular intrusion detection system, named VIDS, that includes two parts: a lightweight domain-based model and a crossdomain-based model. The domain-based model utilizes LSTM, a Recurrent Neural Network (RNN), and takes the time frequency difference between CAN messages as input. The crossdomain-model uses the data field (64-bit) values in each CAN message as input for ANN to detect anomalies.

All machine learning-based schemes usually involve high computation costs. Although all proposed IDSs are supposed to work in real-time, only a few of them report an evaluation of the processing delay time. In the work of [49], they report a real-time processing delay of 2.05 ms-3.78 ms per CAN message, depending on the number of layers used in the DNN model. In another work [2], the authors conduct a time complexity analysis on a different number of layers. In their evaluation, the best detection processing time is about 5-6 ms with two hidden layers. With the increase of the hidden layer, the detection can cost up to 20 ms with sixteen layers.

3.2.1 General Machine Learning-based Approach

The application of classical machine learning methods to IDS has been a significant area of focus in cybersecurity research. The basis for this approach is to employ machine learning techniques that enable computers to learn from and make decisions or predictions based on data. Machine learning provides a range of advantages, such as the capability to handle large-scale data, learn and adapt to new patterns, and make predictions with high accuracy [99].

As an example, decision trees have been widely utilized in IDS due to their interpretability and ability to handle both categorical and numerical data. A study by Kumar et al. [56] uses decision tree-based ensembles for network-based intrusion detection, demonstrating their effectiveness. Similarly, Support Vector Machines (SVMs) have been adopted for their ability to handle high-dimensional spaces and their robustness against overfitting. Tao et al. [109] showcase an anomaly detection system using a genetic algorithm and SVMs that detected attacks effectively.

The k-Nearest Neighbor (KNN) algorithm, owing to its simplicity and efficacy in multi-class classification problems, has also found its use in IDS. A study by Rao et al. [10] successfully employs KNN in a network intrusion detection system, highlighting its efficiency. Moreover, clustering algorithms like K-means and density-based spatial clustering of applications with noise (DBSCAN) have also been leveraged to develop IDS. A study by Peng et al. [89] utilizes clustering techniques to detect anomalies in a network traffic dataset, showcasing their potential over big data. In addition, ensemble learning methods have gained popularity, offering improved performance by combining the strengths of multiple learning models. A study by Illy et al. [47] demonstrates the effectiveness of this approach through two levels of detection in network intrusion detection.

Despite the promises shown by classical machine learning techniques in IDS, it is worth noting that these methods often require significant feature engineering and may struggle with dynamic, evolving threats [132]. Nevertheless, applying these techniques to intrusion detection provides a robust foundation upon which more advanced systems can be built.

3.2.2 Neural Network-based Approach

3.2.2.1 Intrusion Detection Using Sequential Patterns

Machine learning techniques have proven to be successful at conducting intrusion detection throughout the years, and intrusion detection using sequential patterns is one of the research directions. Oliveira et al. propose an approach by using sequential patterns of data, and their results show that anomaly detection can be better addressed from a sequential perspective [84]. Xing et al. categorize three types of classification methods based on the sequence pattern of data [131]. The first one is the feature-based method that transforms sequences into feature vectors through feature selections and then applies classification methods. This method introduces an additional preprocessing step before the classification. The second method is the distance-based method that examines the similarity among sequences. This method may not work effectively in environments that involve a large-scale database, as it is difficult to calculate the distances of all reference sequences. The last method utilizes machine learning algorithms, such as naive bayes, markov model, hidden markov model, and neural networks, to build sequence models for sequence classification. However, for high accuracy, they usually involve high computation costs and require large amounts of data in the training process. Wang et al. propose a malicious traffic classification method using CNN [124]. By transforming the raw network traffic data as images for their training and testing of the CNN classifier, there is no necessity to introduce any hand-designed features or feature selections. To protect CAN in particular, with a similar design, a CNN-based IDS is proposed [102]. CAN messages are converted to image-like frames with grid structure rather than feature vectors, so therefore their CNN classifier is able to learn the sequential patterns of CAN traffic to detect intrusions. Their experiments demonstrate that the proposed IDS has low false negative rates and error rates.

3.2.2.2 Intrusion Detection Using Spatial Features

Neural networks have been widely and successfully used for intrusion detection, and spatial feature is one of the commonly used traffic data features [123]. To exploit spatial features, traffic is first

converted into traffic image frames. Subsequently, image classification techniques are employed to learn from these frames. This knowledge is then applied to classify images for intrusion detection. For CAN IDSs, several studies consider spatial features of CAN traffic. A CNN-based IDS approach is proposed [102], where CAN IDs are transformed into image frames rather than feature vectors. However, the proposed IDS is limited to detecting malicious messages with altered data sections. Another CAN IDS provides a design to convert CAN traffic data [138]. The design takes both the ID section and the data section into account to enable the IDS to detect more attack types, such as attack messages that only contain malicious data. In addition, some studies propose that CNN can work with other machine learning approaches, including LSTM [67, 123], whereas those schemes tend to be large in scale and can be problematic when deploying them to specific environments such as embedded systems.

CHAPTER 4

Empirical Comparative Study

Numerous Intrusion Detection System (IDS) schemes are proposed to protect in-vehicle networks. These schemes have been examined through diverse datasets, evaluation metrics, and adversary models. Their evaluations can be limited when depending on simulated data, data from a single vehicle, or considering only a certain type of attack. Moreover, these schemes have shown varied detection accuracy due to the use of different datasets and a narrow focus on specific attack types. Therefore, understanding the effectiveness of a particular IDS can be challenging. To gain a comprehensive understanding, it is crucial to evaluate these schemes under the same conditions, employing the same types of attacks, datasets, and evaluation metrics.

In this chapter, we implement representative existing schemes to provide a comprehensive understanding of CAN IDS. We begin by explaining the adversary model we used. We then detail our data collection. Next, we discuss the evaluation metrics employed in this study. Finally, we conduct a comparative study on rule-based IDS schemes and machine learning-based IDS schemes respectively.

4.1 Adversary Model

We consider general adversary models, which cover most known attack scenarios against CAN communication, in contrast to the limited scope often found in other studies. In this section, we present the adversary models used in this comparative study. We first introduce the various types of

attacks that can be performed against CAN. Following this, we discuss different types of adversaries with different levels of capabilities.

Understanding how adversaries gain access to the vehicle's bus is important to the context. Direct access typically involves a physical connection to the CAN bus, usually through the On-Board Diagnostics-II (OBD-II) port. Such access provides a direct path to the in-vehicle network, making all ECUs potentially exposed [13]. In more severe cases, adversaries can compromise an ECU, potentially controlling the specific subsystem and having the ability to send malicious CAN messages throughout the in-vehicle network [14]. Another type of access is indirect access which occurs remotely, leveraging the vehicle's wireless interfaces. These interfaces range from unsecured or weakly secured Wi-Fi networks to Bluetooth interfaces or even the vehicle's cellular connectivity. Through these channels, adversaries are able to remotely inject malicious CAN messages, initiate unauthorized control commands, or disrupt the normal operations of the in-vehicle network. Whether access to the target CAN bus is achieved directly or indirectly, it paves the way for adversaries to launch a variety of attacks.

4.1.1 Attack Types

Depending on whether the attacker can change (increase or decrease) the original CAN traffic normal volume, we consider the following types of attacks:

Injection Attack: In this attack, the attacker injects malicious CAN messages into the bus, which increases CAN traffic volume. As shown in Figure 4.1 (a), the attacker injects malicious messages (in red) that increase the rate of the CAN messages so that it is higher than normal.

There is no need to compromise an ECU to launch this type of attack, and the attacker may send CAN messages directly to the bus via an extra ECU. Specifically, based on the content of CAN messages, attacks of this type can be further classified as follows:

- *Random ID Attack*: In this attack, the attacker generates random CAN messages and injects them into the bus. The attacker does not need to have any prior knowledge of the target vehicle.

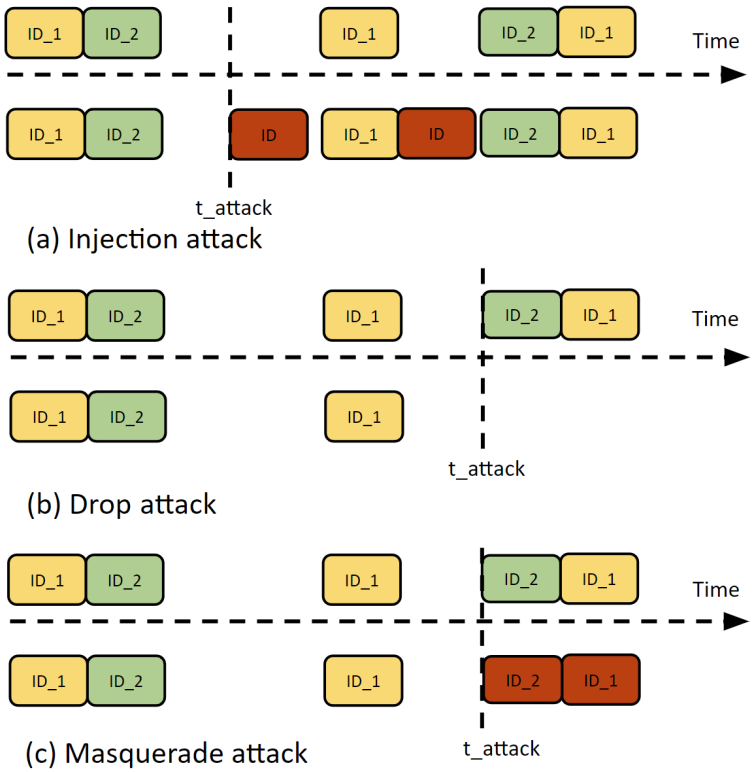

Figure 4.1: Types of attacks: (a) injection attack, (b) drop attack, and (c) masquerade attack

- *All Zero ID Attack*: In this attack, the attacker injects messages with the ID set to zero. Since a lower ID value represents a higher priority, a zero ID message has the highest priority. The attacker's goal is to occupy the bus to a certain extent such that it can cause the DoS effect for other ECUs, which may fail to function, resulting in some severe safety consequences.

- *Replay Attack*: In this attack, the attacker injects messages that he has seen before. The attacker collects CAN messages transmitted on the bus and replays them later. Because CAN lacks freshness protection mechanisms, the attacker can implement this attack easily.

- *Spoofing Attack*: In this attack, the attacker injects spoofed messages pretending to be from a valid ECU. To create a valid spoofed message, the attacker needs to know the format of messages sent from the target ECU. The attacker can acquire such knowledge by looking at the proprietary DBC file (CAN database) to see if it is accessible or through reverse engineering,

as CAN messages are transmitted in plaintext. This attack can cause the ECU to malfunction.

When an attacker seeks to control the vehicle, they can transmit messages with specific IDs to the bus. As a result, the targeted ECU will receive messages from both the legitimate ECU and the attacker. To ensure the malicious messages take precedence over the authentic ones, the attacker has to transmit them at a faster rate than the original ECU. While a typical message injection attack might involve the attacker sending twice as many messages as the normal rate, in real-world scenarios, the volume of malicious messages can surge, sometimes being 20 to 100 times more than the regular messages [72].

Drop Attack: In this attack, the attacker drops messages that are supposed to be transmitted, decreasing the overall CAN traffic volume. As shown in Figure 4.1 (b), the attacker drops multiple normal messages (in green and yellow) that decrease the overall volume of the CAN traffic. The attacker needs to control a compromised ECU, and then the attacker is able to stop or suspend selected or all CAN messages which should be sent to the bus by manipulating the compromised ECU. As a result, the compromised ECU looks like it is "disconnected" from the bus, which can result in serious errors for the attacked vehicle.

Masquerade Attack: In this attack, the attacker sends masqueraded CAN messages to replace the real ones, as shown in Figure 4.1 (c). This attack will not change the CAN traffic volume on the bus. The attacker needs to compromise an ECU as the prerequisite. The attacker can execute this attack by initiating a drop attack to disable an ECU, followed by injecting spoofed messages impersonating the disabled ECU, or by directly manipulating the input/output of a compromised ECU to send CAN messages with malicious content.

4.1.2 Adversary Types

We consider that an adversary can have three different levels of capabilities.:

- *Weak*: A weak adversary has no idea about the semantic meaning of CAN messages of the target vehicle and has no possession of previous CAN traffic traces. So a weak adversary can

only send random and simple messages. The objective of the weak attacker is to compromise the usability of normal messages or to win the arbitration. A weak adversary can typically execute the random ID attack and the all zero ID attack.

- *Medium*: A medium adversary has all the capabilities of a weak adversary. In addition, it can have access to current or previous traffic traces, have knowledge about the specification of certain CAN IDs either through reverse engineering or learning from other sources such as the proprietary DBC file, and have access to a compromised ECU to drop certain CAN messages which should be sent. The objective of the medium adversary is to harass the CAN bus or disturb the vehicle's behaviors. A medium attacker can launch all types of injection attacks, including the replay attack and the spoofing attack, and the drop attack.

- *Strong*: A strong adversary possesses all the capabilities of a medium adversary, with the added ability to execute masquerade attacks. To achieve this, the attacker must analyze the CAN message traffic in-depth, focusing on aspects like message frequency distribution. This attack does not alter the usual volume of CAN traffic.

4.2 Data Collection

Although the intrusion detection research field has standard benchmark datasets, like the popular KDD'99 and its related sub-dataset NSL-KDD [110], a significant gap persists with the lack of standard benchmarks for in-vehicle network IDSs. While many researchers rely on simulated data, utilizing actual in-vehicle data is crucial for accurately evaluating proposed schemes. Some evaluations are conducted using data from a single vehicle or only a small number of CAN messages are collected within a few minutes, making claims of their effectiveness questionable. Furthermore, it is uncertain whether these schemes can deliver comparable performance in different in-vehicle environments due to variations in CAN implementation and setup across manufacturers.

To address these issues and facilitate systematic evaluations, we collect datasets from a variety of vehicle models and production years. It is important to note that all vehicles used in this study

are unmodified, serial production vehicles with valid licenses. In this section, we first outline our data collection setup, followed by an overview and discussion of the datasets collected from seven real vehicles.

4.2.1 Equipment and Routes

To collect data, we use the CANalyst-II device that supports different protocols, including CAN, CANopen, and SAE J1939 protocol. This CAN analyzer collects CAN data via the on-board diagnostics II (OBD-II) port. As an on-board diagnostic standard, OBD-II port is mandatory for all vehicles sold in the US. Although it is mainly used for diagnostic and emissions measurements, it also provides additional information, including engine control, body control, and chassis control information. The OBD-II port has 16-pin, and it is typically located on the left side of the driver's dashboard, near the steering column. To collect data, the analyzer must be connected to a laptop with the supporting software installed through a USB port. Then, the CAN Analyzer is plugged into the OBD-II port of the vehicle, which is operating normally.

We collect datasets from seven real vehicles of various models from different manufacturers. CAN messages are collected under normal operations for 20-40 minutes. For example, we collect datasets from a 2018 Honda Civic during four round trips on a road segment on campus, as shown in Figure 4.2, all under normal operations.

4.2.2 Datasets and Message Characterization

Details of the datasets are presented in Table 4.1. The second column represents the total number of messages in each dataset. The third column indicates the number of unique message IDs, varying by manufacturer and vehicle model. The fourth column provides the number of periodic messages.

Many CAN messages with specific IDs are sent to the bus at regular intervals. This periodicity feature plays an important role in CAN IDS design. Monitoring the periodicity of messages is one of the key detection features. This feature is widely exploited by a range of existing schemes, including both rule-based and machine learning-based schemes. We notice two datasets contain

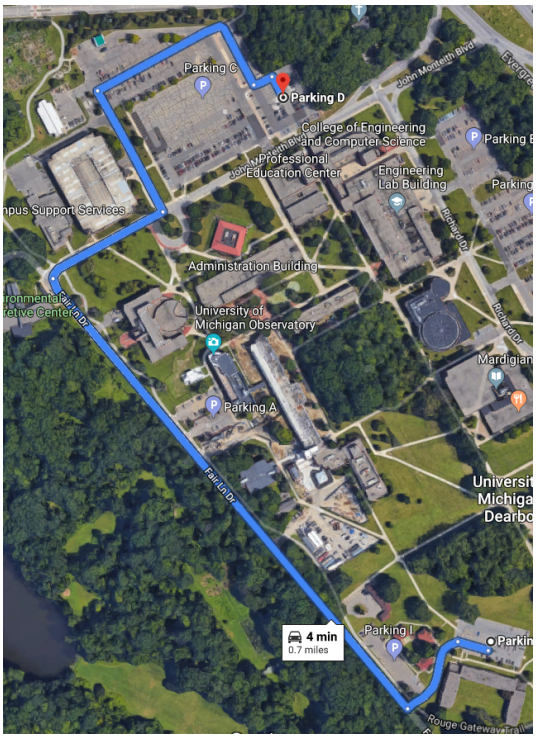

Figure 4.2: Driving path of data collection

aperiodic messages: 2019 Chrysler Pacifica and 2013 Ford Fusion. In the 2019 Chrysler Pacifica, there are 173 distinct IDs: 132 of them are sent periodically at intervals ranging from 1 ms to 2 seconds, and 41 of them are sent aperiodically. In the 2013 Ford Fusion, there are 79 distinct IDs: 70 of them are sent periodically at intervals ranging from 20 ms to 1 minute, and 9 of them are sent aperiodically. All messages are sent periodically in the other five datasets.

The normal distribution is considered an appropriate criterion for determining whether a message is periodic [20]. When messages with specific IDs are transmitted periodically, the time intervals between them should adhere to a normal distribution. For example, as demonstrated in Figure 4.3, the distribution of time intervals for the example ID 0x224 aligns well with a normal distribution.

To classify a message as periodic, we utilize standard deviation and the coefficient of variation. We have a set of time intervals for a unique ID, denoted as $T = \{T_1, \cdots, T_n\}$, where T_i is the i_{th} interval in the list, and n represents the total number of values in the list. We calculate the mean $\overline{T}$, the standard deviation σ, and the coefficient of variation c_v of these time intervals as follows:

Table 4.1: Details of datasets

Name of Dataset	Number of Messages	Number of Unique IDs	Number of Periodic Messages
Chevrolet Volt 2013	4,536,342	106	All
Chrysler Pacifica 2019	2,470,310	173	132
Ford Fusion 2006	808,423	17	All
Ford Fusion 2007	710,244	20	All
Ford Fusion 2013	2,158,201	79	70
Honda Accord 2006	243,762	13	All
Honda Civic 2018	1,806,780	54	All

$$\overline{T} = \frac{\sum_{i=1}^{n} T_i}{n} = \frac{T_1 + T_2 + \cdots + T_n}{n} \tag{4.1}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} \left(T_i - \overline{T}\right)^2}{n-1}} \tag{4.2}$$

$$c_v = \frac{\sigma}{\overline{T}} \tag{4.3}$$

We use the coefficient of variation, represented as c_v, to determine periodicity. This measure is specifically used to understand if messages associated with a certain ID are sent at regular intervals. The c_v presents a relative estimation of the standard deviation, thereby providing a measure of variability relative to the mean.

When an ID's c_v value falls below a predetermined threshold, it is labeled periodic. The threshold value for defining periodicity should be set carefully; it is usually established between

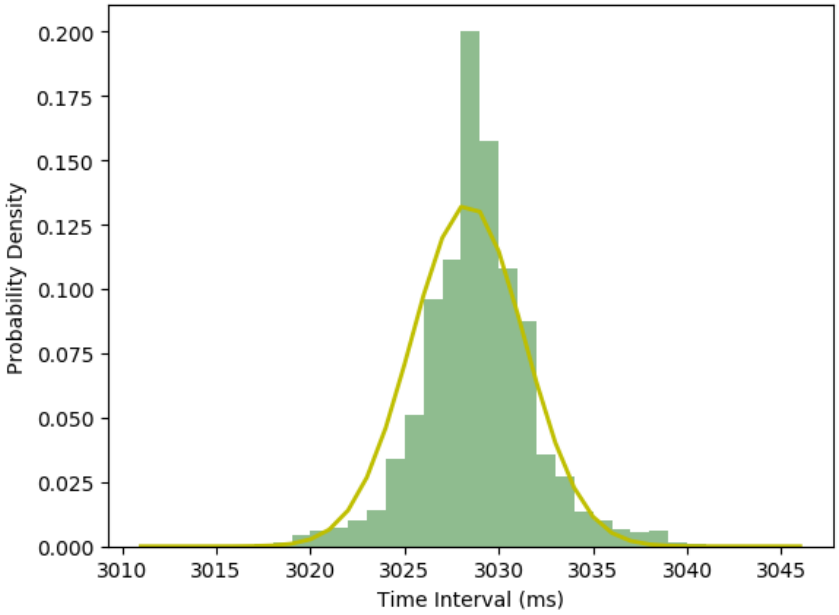

Figure 4.3: Distribution of time intervals for ID 0x224

10% and 20%, according to standard practice in the field. In our experiments, the threshold value of 20% is selected with consideration of the distribution of all c_v values present in datasets.

4.3 Evaluation Metrics

To provide a clear comparison among various IDSs, it is critical to apply the same evaluation metrics consistently across all schemes. This strategy eliminates potential biases or discrepancies that may arise from the use of different evaluation standards, thereby allowing for a more objective and direct comparison. In this section, we discuss the specific evaluation metrics employed in our study to assess the effectiveness and efficiency of the proposed schemes. Furthermore, we introduce the computation time, a metric used to examine efficiency, which is crucial in the context of real-time in-vehicle networks. These metrics provide a standardized measure that allows for consistent comparisons across different schemes.

4.3.1 Effectiveness

To comprehensively evaluate the effectiveness of IDS schemes, we employ a suite of scientific metrics that include accuracy, true positive rate (TPR, detection rate, also known as sensitivity or recall), true negative rate (TNR, or specificity), false negative rate (FNR), false positive rate (FPR), and the F-1 score.

Accuracy is often used in performance evaluation, which quantifies the proportion of total predictions that are correct. High accuracy is indicative of a model with remarkable overall predictive capability. TPR, a critical measure of the system's proficiency in correctly identifying malicious instances, should ideally be maximized. Conversely, TNR illustrates the system's ability to accurately identify benign or non-threatening instances. A high TNR ensures minimal false alarms. In contrast, FNR quantifies the fraction of actual undetected threats. A low FNR signifies the system's efficacy in threat detection. FPR is the ratio of normal instances wrongly identified as threats, which we strive to minimize to avoid unnecessary panic and resource allocation. Finally, the F-1 score is a harmonic mean of precision and recall. This score serves as a robust performance measure, especially for imbalanced datasets, encapsulating both the system's precision (low FPR) and its sensitivity (high TPR). It is a crucial benchmark for comparing the performance of different IDSs. By using these various metrics, we aim to provide a comprehensive evaluation of the performance of IDS, highlighting areas of potential enhancement.

Table 4.2: Confusion matrix

	Actually Malicious	Actually Normal
Detected Malicious	TP	FP
Detected Normal	FN	TN

To calculate them, we have a confusion matrix, as shown in Table 4.2, and the following terms

are defined:

- *TP (true positive)*: the number of malicious messages that are correctly detected.
- *FP (false positive)*: the number of normal messages that are incorrectly detected.
- *TN (true negative)*: the number of normal messages that are correctly detected.
- *FN (false negative)*: the number of malicious messages that are incorrectly detected.

Accuracy, TPR, TNR, FNR, FPR, and F1-Score are calculated as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.4}$$

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.5}$$

$$\text{TNR} = \frac{\text{TN}}{\text{FP} + \text{TN}} \tag{4.6}$$

$$\text{FNR} = \frac{\text{FN}}{\text{TP} + \text{FN}} \tag{4.7}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{4.8}$$

$$\text{F1-Score} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(FP + FN))} \tag{4.9}$$

An effective IDS should achieve a high detection rate and a low false alarm rate. A low false positive rate is critical to maintaining system credibility, as error-prone systems will gradually lose trust and their alarms, whether true or false, will be ignored. This is particularly relevant in the context of in-vehicle networks [14], where false alarms can also lead to unnecessary panic or waste resources such as emergency services. Moreover, false alarms can potentially trigger a cascade of resource-draining responses, both in terms of personnel and system resources.

4.3.2 Computation Time

In the context of our study, we use the terms "detection latency" and "processing delay" interchangeably to refer to the computational time. It is the time required to process inputs by an IDS. This metric holds crucial importance in real-time detection. For IDS based on machine learning, we discuss the computation time taken for predicting or classifying data instances in a given input frame utilizing a specific machine learning model. This involves the time from when the model first gets input to when it outputs the final prediction.

It is noteworthy that this latency or delay is an essential factor in determining the efficiency of the IDS, particularly in a real-time system where timely anomaly detection can be critical. Longer detection latencies mean that the system might fail to react to an intrusion in a timely manner, potentially leading to significant security risks. Therefore, the efficiency of machine learning models in this context is not only determined by their accuracy but also by how quickly they can process and classify inputs.

4.4 Rule-based IDS Comparative Study

There are two main categories in the field of IDSs for CAN: rule-based IDS and machine learning-based IDS. In this section, to gain a comprehensive understanding of rule-based IDSs, we conduct a comparative study. We implement and evaluate representative rule-based IDSs.

In this study, the evaluation of these schemes is conducted under the same adversary model, using the same evaluation metrics, platform, and datasets to provide an in-depth analysis of the comparative performance and effectiveness of these rule-based IDSs. A key feature of our study is the use of data acquired from real vehicles, enhancing the practical relevance of our findings.

Generally, rule-baed IDSs construct a standard normal model using statistical methodologies or leveraging the characteristics of CAN data as "rules", which serve as simple and straightforward detection patterns. Various features extracted from in-vehicle network traffic, such as message frequency, message interval, statistics, entropy, message data section, and sensor data, are utilized to develop the normal model. Rule-based IDSs possess certain advantages, including lower computational overhead and quick response times. However, they also have limitations. The primary drawback of rule-based IDSs is their relatively low accuracy, as it proves challenging to detect all types of attacks by relying on a single rule or even a collection of rules.

4.4.1 Summary of Rule-based IDS

Rule-based IDS schemes are summarized in Table 4.3. A key observation from these schemes is their general lack of utilization of all CAN message data. Additionally, they tend to target a single type of attack and apply a rather limited dataset, which can narrow their applicability. In the following, we extend our discussion on these schemes, exploring their limitations in greater detail.

This section clarifies the methodologies each scheme employs and engages in a deeper discussion about their specific strengths and limitations. It is important to note that the specific "injection attack" listed in the table does not include every type of injection attack discussed in Section 4.1. The focus is on highlighting the action of launching an attack by injecting malicious messages.

Table 4.3: Summary of rule-based IDS

Detection Feature	References	ID /Data Section	Attacks Detected	Evaluation Datasets
Message Frequency	[41] [65]	ID	Injection Attack	Simulated Data Simulated Data
Message Interval	[101] [74] [25]	ID	Injection Attack	One Vehicle Data One Vehicle Data No Evaluation
Message Entropy	[76] [69]	ID	Injection Attack	One Vehicle Data One Vehicle Data
Message Sequence	[68]	ID	Injection Attack	One Vehicle Data
Accumulated Sum	[14] [87]	ID	Injection /Drop/Masquerade Attack Injection Attack	Three Vehicles Data One Vehicle Data
Hamming Distance	[103]	ID + Data	Injection Attack	One Vehicle Data

Frequency-based IDS: This IDS uses message frequency as the detection feature. CAN messages of a given ID are often broadcast by a single ECU and in a constant frequency. That is, the rate of certain ID messages transmitted can keep a relatively constant value. In [41], authors first generate a baseline from the normal traffic of the bus. Their IDS can monitor the bus to count the rate of current messages transmitted and compare the count result with the baseline. Any missing or additional messages out of the threshold can trigger an alarm. In [65], authors measured anomalies observed in traffic frequency to detect injection attacks. The alarm threshold is calculated based on the ID transmitted on the bus with its uninterruptible occurrence frequency. In [72], their scheme monitors the rate of the CAN messages. When the rate is higher than normal, an alarm will be triggered.

Even though frequency-based IDS schemes have the potential to provide fast response on embedded CAN controllers with limited resources, they are unable to detect certain types of attacks, such as the masquerade attack, as it does not change the frequency by withholding valid messages and sending messages with valid ID and modified data section. In addition, they are not sensitive to the injection attack and the drop attack with a small number of messages. Furthermore,

they cannot handle messages which are sent aperiodically to the bus.

Interval-based IDS: This IDS uses message intervals as the detection feature. As CAN messages with the same ID are usually sent to the bus periodically, time intervals between consecutive messages with the same IDs should be relatively steady. Interval-based approaches utilize this fact to build baselines and detect attacks. They differ from frequency-based IDSs as they count the timing intervals of the messages as opposed to the transmission rate of messages. Song et al. [101] propose a scheme that can examine the time interval of CAN messages for anomaly detection. The time interval feature is capable of detecting message injection attacks because of the difference between the time intervals of messages in normal status and attack status. The scheme proposed in [25] has the same core design, and both of them count a message as an attack message when the time interval is below half of the normal interval. Authors of [74] propose simple intrusion detection systems based on the message interval. To set the threshold, for n messages with a given ID, denote time stamps for that ID, t_0, t_1, $\cdots$, t_n. The time intervals are: $\triangle_1 = t_1$ - t_0, $\cdots$, $\triangle_n = t_n$ - $t_{(n-1)}$. Authors use the maximum observed error from expectation and have $m = max_i|\triangle_i - \mu|$, where $\mu = \sum_i(\triangle_i/\text{n})$. The threshold of the scheme is set as m + (15% * μ) for absolute error from the expectation.

The strength of these detection algorithms is that they are simple to use. However, they are not able to handle the masquerade attack as well as injection and drop attacks with a small number of malicious messages. Moreover, they cannot deal with messages which are sent aperiodic to the bus.

Entropy-based IDS: This IDS uses message entropy as the detection feature. Entropy-based IDS schemes characterized the normal behavior of a set of normal CAN traffic traces based on the level of the statistical entropy. The values of entropy calculated from normal messages keep stable statistical characteristics, while attacks can introduce significant deviations and break the stable status. Those schemes observed the bus information entropy in accordance with the fixed time window and calculated entropy for CAN message ID according to Shannon entropy. Assuming system Z has a limited set of possible states: z_1, z_2, $\cdots$, z_n, and the information entropy of system

Z is

$$E(Z) = -\sum_{k=1}^{n} p(z_i) log(p(z_i)) \tag{4.10}$$

where $p(z_i)$ is the probability of system Z in state z_i. Muter and Asaj [76] introduce the concept of entropy-based detection for in-vehicle networks for the first time. They focus on groups of messages with the same ID instead of overall CAN traffic flow, which adds benefit to detecting ID-specific anomalies. The limitation of their approach is its inability on detecting a small number of injected CAN messages. In another entropy-based scheme [69], authors evaluate the effectiveness of entropy-based schemes in [76] with complete real vehicle data, and they found that the direct use of entropy-based IDS for anomaly detection is only effective in the case of a large number of malicious CAN messages.

Sequence-based IDS: This IDS uses message sequence as the detection feature. In [68], the proposed scheme can identify anomalies in the sequence of messages that are transmitted on the CAN bus based on the ID sequence between messages. The main idea is to build a valid ID sequence list based on a sequence feature that is based on recurring patterns within the sequence of message IDs of the normal CAN traffic trace. The authors state that every ID in their test model is followed only by a subset of all the available IDs, thus there are limited admissible transitions between all IDs. By using this fact, the valid list order of the ID sequence is built by recording distinct pairs of IDs of every two consecutive CAN messages from the legit traffic trace. To detect anomaly behaviors, the scheme monitors the ID sequence of the current traffic, and an alarm can be triggered when an ID sequence appears but is not in the normal list. Since the scheme only covers the order of two consecutive IDs (not a sequence chain), one of the limitations is that the proposed scheme cannot detect the replay attack and the masquerade attack. Another limitation is that only the ID part is monitored. Any malicious messages with a modified data section cannot be detected.

Accumulated sum-based IDS: This IDS uses accumulated sum as the detection feature. The accumulated sum [14, 87] is derived from the periodicity feature of CAN messages and can be

counted as an extension of the interval-based feature. It cannot detect the aperiodic messages as well.

Hamming distance-based IDS: This IDS uses Hamming distance as the detection feature. The work of [103] calculates the Hamming distance of data sections between sequences messages with the same given ID. The Hamming distance between two binary strings of K-bits is calculated as:

$$H(x, y) = \sum_{i=1}^{k} |x_i - y_i| \tag{4.11}$$

In this equation, x and y are two data sections in two consecutive messages with the same ID. To build a baseline, they extracted the minimum and maximum values of the Hamming distance for each distinct ID transmitted on the bus. Anomalies can be identified based on a significant deviation from the calculated Hamming distance function during the detection process. However, this scheme cannot detect the replay attack, and it is not sensitive to messages that have no strong patterns of Hamming distance values.

4.4.2 Experimental Results

In our experiments, we implement and assess selected representative IDS schemes on a Linux-based system running on Ubuntu 20.04.2 LTS. This system is powered by an AMD Ryzen 3700x CPU, equipped with 8 cores and 16 threads, with a base clock speed of 3.6 GHz.

The following metrics are used: true positive rate (TPR, or detection rate), false positive rate (TFR), false positive rate (FPR), false negative rate (FNR), and accuracy as defined in Section 4.3. For the adversary model, an attacker is able to perform various forms of attacks, including injection attack, masquerade attack, and drop attack, as discussed in Section 4.1. Moreover, we also consider hybrid attacks, where the attacker can launch each type of attack randomly.

We utilize data collected from real cars, as described in Section 4.2. Based on thorough analysis, we carefully select a representative dataset, the Honda Civic dataset. Since our adversary model can randomly launch different types of attacks, the results are average values of 10 repeated experiment

results. Experimental results are provided in Table 4.4.

Table 4.4: Experimental results for rule-based IDS

Injection Attack	TPR	TNR	FPR	FNR	Accuracy
[41]	98.42%	93.27%	6.73%	1.58%	96.43%
[65]	97.91%	91.07%	8.93%	2.09%	95.27%
[101]	98.82%	91.76%	8.24%	1.18%	96.09%
[74]	92.55%	90.85%	9.15%	7.45%	91.89%
[25]	97.07%	93.33%	6.67%	2.93%	95.63%
[76]	70.34%	72.04%	27.96%	29.66%	71.46%
[69]	67.24%	71.40%	28.60%	32.76%	70.13%
[68]	60.16%	72.73%	27.27%	39.84%	68.79%
[14]	95.02%	96.25%	3.75%	4.98%	95.91%
[87]	95.29%	96.31%	3.69%	4.71%	96.02%
[103]	66.67%	74.91%	25.09%	33.33%	72.66%
Drop Attack	TPR	TNR	FPR	FNR	Accuracy
[41]	83.33%	87.50%	12.50%	16.67%	86.38%
[65]	0%	0%	0%	0%	0%
[101]	0%	0%	0%	0%	0%
[74]	0%	0%	0%	0%	0%
[25]	0%	0%	0%	0%	0%
[76]	68.72%	63.52%	36.48%	31.28%	65.67%
[69]	64.05%	67.67%	32.33%	35.95%	66.18%
[68]	57.55%	69.74%	30.26%	42.45%	62.26%
[14]	96.77%	91.09%	8.91%	3.23%	94.58%
[87]	96.56%	90.50%	9.50%	3.44%	94.22%
[103]	67.88%	69.28%	30.72%	32.12%	68.52%
Masquerade Attack	TPR	TNR	FPR	FNR	Accuracy
[41]	0%	0%	0%	0%	0%
[65]	0%	0%	0%	0%	0%
[101]	0%	0%	0%	0%	0%
[74]	0%	0%	0%	0%	0%
[25]	0%	0%	0%	0%	0%
[76]	0%	0%	0%	0%	0%
[69]	0%	0%	0%	0%	0%
[68]	0%	0%	0%	0%	0%
[14]	75.26%	64.32%	35.68%	24.74%	71.04%
[87]	76.53%	61.41%	38.59%	23.47%	70.69%

[103]	60.00%	67.94%	32.06%	40.00%	63.06%
Hybrid Attack	TPR	TNR	FPR	FNR	Accuracy
[41]	79.28%	86.56%	13.44%	20.72%	82.09%
[65]	68.82%	61.56%	38.44%	31.18%	66.02%
[101]	75.81%	60.35%	39.65%	24.19%	69.85%
[74]	72.52%	65.04%	34.96%	27.48%	69.63%
[25]	68.46%	62.87%	37.13%	31.54%	66.31%
[76]	62.17%	56.67%	43.33%	37.83%	60.05%
[69]	67.80%	53.62%	46.38%	32.20%	62.33%
[68]	53.81%	65.80%	34.20%	46.19%	58.44%
[14]	87.62%	93.05%	6.95%	12.38%	89.71%
[87]	87.90%	89.55%	10.45%	12.10%	88.54%
[103]	58.58%	61.96%	38.04%	41.42%	59.88%

Those existing schemes generally do not perform as well as reported in their studies. The main reasons are as follows: some IDS schemes are not explicitly designed to detect certain types of attacks. Furthermore, these schemes' effectiveness can be compromised, as the threshold is often determined based on their own adversary models. Considering that real attack scenarios can be more complex, this can lead to IDS not being robust enough. Moreover, there is an issue of sensitivity. Some IDSs may not be adequately sensitive to a small number of attack messages, which can allow minor but potentially harmful activities to go unnoticed. This highlights the importance of designing effective IDS schemes that can effectively detect and counter a wide range of attacks.

In conclusion, while there exists a wide range of rule-based IDS strategies developed by researchers, it is clear that a number of these systems may be under-equipped to handle more complicated attack scenarios. The adversary models considered in their designs typically only involve a smaller range of attack types, which may weaken their effectiveness in real-world environments where threats are often more sophisticated and diverse. It is, therefore, essential to consider these limitations in the ongoing development and evaluation of IDS schemes.

It can also be noticed that, when dealing with real-world attacks, relying on a single rule is often not enough. This leads us to consider how to incorporate multiple rules or combine these rules

with other complementary techniques to protect in-vehicle networks efficiently. This combination of strategies has the potential to greatly improve intrusion detection systems, making them better able to defend against a wide variety of advanced cyber threats.

4.5 Machine Learning-based IDS Comparative Study

A common challenge for CAN IDS is the lack of standard benchmark datasets, resulting in researchers using various datasets, including simulated data, to train and test their IDSs. Such datasets for training and testing are often limited. Besides, the evaluation of different IDS schemes often involves a diverse array of performance metrics, implemented on various platforms, and tested under different adversary models. As a result, directly comparing different solutions becomes challenging. It is unclear whether differences in performance are due to the IDS design itself, or simply a reflection of the specific factors, such as characteristics of the datasets used. It is critical to assess these schemes using consistent performance metrics, on the same platform, and under the same adversary model. In this section, we perform a comparison study to investigate various machine learning-based IDSs for CAN, aiming to provide a comprehensive understanding of these IDSs.

For machine learning-based IDS schemes, similar issues exist with rule-based IDSs. Fundamental aspects such as the training and testing datasets used, the type of adversary model considered, and the specific evaluation metrics affect the performance evaluation. In this comparative study, we aim to better understand the relative strengths and weaknesses of these IDS schemes. To facilitate a comprehensive understanding and fair comparison of these machine learning-based IDSs, we implement these systems using the same datasets and evaluation metrics, and deploying the same adversary models, which can provide a more accurate insight into those schemes.

4.5.1 Summary of Machine Learning-based IDS

A machine learning-based IDS schemes usually consist of two steps: training and detection. In the training step, a labeled CAN dataset is used for training the machine learning algorithm. This step

outputs a trained model that serves as a classifier for detection. In the detection step, the classifier monitors the real-time CAN traffic and determines whether a message is malicious or not. An alarm can be triggered if a malicious message is detected.

We summarize the representative machine learning-based IDSs in Table 4.5. Most of these systems use Identifier (ID) sections of CAN messages for their training and testing. Some studies opt to use OBD-II Parameter IDs (PIDs) rather than CAN data [111, 78, 125]. OBD-II PIDs, defined by the SAE standard J1979, are codes used to request specific vehicle data and are primarily used for diagnostic purposes. These codes are mandated for support by on-road vehicles sold in North America, mainly for emission inspections. By utilizing PIDs, the datasets in these studies have explicit semantic meanings related to emission/control systems. Those machine learning-based IDSs typically focus on identifying injection attacks and are often evaluated using simulated data or data from a single vehicle, limiting their applicability across a diverse range of vehicles and attack types.

In the following, we present details of each scheme and discuss strengths and weaknesses.

Table 4.5: Summary of machine learning-based IDS

Type	**References**	**ID /Data Section**	**Attacks Detected**	**Evaluation Datasets**
HMM	[78]	PID	Injection Attack	Three Vehicle Data
SVM	[111]	PID	Injection Attack	Simulated Data
Decision Tree	[113]	ID + Data Entropy	Injection Attack	One Vehicle Data
ANN	[125]	PID	Injection Attack	One Vehicle Data
DNN	[49]	Data	Injection Attack	Simulated Data
LSTM (RNN)	[112]	ID	Injection Attack	One Vehicle Data
CNN	[102]	ID	Injection Attack	One Vehicle Data

HMM-based IDS: In [78], a hidden Markov Model-based technique is employed to build an IDS that monitors current CAN traffic and detects anomaly behaviors based on the deviation from

the messages sequence. This scheme is based on the assumption that the movement of a vehicle is a sequence of states that are dependent on the previous ones. To pre-possess the input data for training, they build time series observations by converting certain CAN messages into meaningful values, including engine RPM, engine coolant temperature, and speed. To avoid a data model that can result in many false positives, authors use gradients for the data model instead of direct observation in the form of real value. The data model is used to construct an HMM model, including the transition and emission probabilities based on sequences of observations. For anomaly detection, they use a sliding window that moves every time when a new observation is available. An anomaly can be caught when the posterior probability of a given prior input sequence is lower than the value of the threshold in the determined sliding window. Even though this scheme used three datasets from three vehicles (Honda Accord, Toyota Corolla, and Chevrolet Cruze) for evaluation, a clear comparison among different datasets is not provided. Besides, details of datasets, such as each vehicle model's year and the length of each dataset, are not provided. The scheme needs further analysis and evaluation by comparing the performance of different vehicles as well as realistic CAN attack messages.

SVM-based IDS: In [111], authors propose an IDS by using one-class Support Vector Machines (SVM) to classify CAN data. Based on analysis of normal CAN data, they measure inter-message timing over a one-second sliding window and obtain statistical values that can represent normal data patterns. The essence of this scheme is to compare the statistical values of current CAN traffic with historical values. A one-class SVM can detect anomalies by using those statistical values. However, this scheme can lack the ability to catch a small number of malicious messages.

Decision Tree-based IDS: Decision tree (DT)-based approaches can handle binary classification tasks. DT needs a supervised labeled dataset during the training stage to be able to make decisions. In [113], a regression DT with Gradient Boosting (GBDT) technique is applied to provide a better IDS. This decision tree method is trained with labeled CAN messages to classify real-time CAN messages into two classes: normal and malicious. The entropy of CAN ID and data are used to construct the decision algorithm. By adding the Gradient Boosting technique, the proposed IDS

can achieve an optimal decision tree model by comparing multiple trained tree models.

Neural Network-based IDS: In [125], the authors propose an ANN-based IDS that uses extracted information from raw CAN messages with PIDs. The information includes five parameters from each feature, torque, Revolutions Per Minute (RPM), and speed, to classify the engine's power ECU behavior in diverse CAN traffic conditions. The training process involves feeding the CAN input data into a bottleneck of the ANN model and learning them via backpropagation. In the detection process, using root-mean-square error on the final output of ANN, an anomaly score is produced and used to determine whether a CAN message is normal or malicious. Despite the promising results in getting higher true-positive detection against false-positive rate, the proposed scheme only uses CAN messages that can be translated by PIDs. Thus, it cannot protect the whole CAN bus.

In [49], authors propose a machine learning-based IDS for the CAN bus network. The classifier is built by a DNN method. To train the DNN, they use extracted information from CAN messages instead of using CAN messages directly. The information is the probability for every CAN message in the form of logistic values "1" and "0" that can separate the normal from the malicious messages. Offline training is performed during the training phase to reduce time consumption, while the trained classifier makes the binary decision for each incoming new CAN message in the detection phase. The authors validate the model by using simulated data [9]. It is reported that the detection rate is 99%. However, training time and testing time increase as more layers are added. And a clear analysis of detection time is not provided.

In [112], a new IDS is proposed by using a special Recurrent Neural Network (RNN) - Long Short-Term Memory (LSTM) to detect attacks on the CAN bus. Their approach works on raw CAN bus data without the need to reduce and extract data from the pre-processing. In [102], authors utilize Convolutional Neural Network (CNN) to build an IDS which is able to detect sequential patterns of vehicle traffic to detect the spoofing attack and the DoS attack. This scheme uses CAN messages directly to build the model without the need for pre-processing. They test their method in an experimental setting and admit that it is difficult to implement in current vehicles.

4.5.2 Experimental Results

To evaluate the aforementioned schemes, we implement them on the same platform used in Section 4.4.2. We employ Python version 3 for implementation and use TensorFlow framework to implement various machine learning algorithms.

We also employ the same adversary model for evaluation. Most machine learning-based IDSs struggle with drop attacks; hence, we consider three types of attacks: injection attack, masquerade attack, and hybrid attack. For consistency, we use the same dataset as used in Section 4.4.2. Results of the experiments are shown in Table 4.6. We use 70% of the data for training and 30% of the data for testing. Due to the stochastic nature of machine learning algorithms, the results can differ across experiments, and we conduct each experiment 10 times and use the average outcome as the final result.

Table 4.6: Experimental results for machine learning-based IDS

Injection Attack	TPR	TNR	FPR	FNR	Accuracy
[78]	79.97%	74.70%	25.30%	20.03%	77.93%
[111]	81.31%	85.62%	14.38%	18.69%	82.97%
[113]	84.19%	71.64%	28.35%	15.82%	79.35%
[125]	96.50%	98.49%	1.51%	3.50%	97.27%
[49]	94.26%	98.64%	1.36%	5.74%	95.95%
[112]	97.04%	99.85%	0.15%	2.96%	98.13%
[102]	86.17%	94.40%	5.60%	13.83%	89.34%
Masquerade Attack	TPR	TNR	FPR	NPR	Accuracy
[78]	66.33%	71.59%	28.41%	33.67%	68.36%
[111]	72.54%	68.10%	31.90%	27.46%	70.82%
[113]	78.14%	75.86%	24.14%	21.86%	77.26%
[125]	89.74%	74.05%	25.95%	10.26%	83.69%

[49]	97.27%	79.19%	20.81%	2.73%	90.29%
[112]	0%	0%	0%	0%	0%
[102]	0%	0%	0%	0%	0%
Hybrid Attack	TPR	TNR	FPR	FNR	Accuracy
[78]	71.15%	75.21%	24.79%	28.85%	72.72%
[111]	83.75%	71.05%	28.95%	16.25%	78.85%
[113]	88.12%	68.36%	31.64%	11.88%	80.50%
[125]	97.37%	81.54%	18.46%	2.63%	91.26%
[49]	97.57%	85.60%	14.40%	2.43%	92.95%
[112]	87.77%	82.28%	17.72%	12.23%	85.65%
[102]	82.74%	85.96%	14.04%	17.26%	83.98%

Based on our experiments, it can be observed that the performance of neural network-based IDS schemes is generally better than those of traditional machine learning methods. Nevertheless, these approaches can involve high false-positive rates, which can hinder their effectiveness in a real-world scenario. In general, these schemes fail to replicate the impressive results presented in their papers. This is primarily because we consider that a general adversary can mount more sophisticated attacks than they. Notably, it can be noticed that the methods proposed in [112, 102] fail to detect the masquerade attack, mainly because their designs solely rely on identifying the ID of CAN messages. Thus, to achieve higher accuracy and robustness against sophisticated attacks, the design of an IDS should consider and leverage both the ID and data sections of CAN messages.

In addition, when compared to rule-based approaches, machine learning methods demonstrate more efficiency in detecting various types of attacks, further validating their potential to enhance the security of in-vehicle networks.

CHAPTER 5

A Hybrid Approach Toward Efficient and Accurate Intrusion Detection for In-Vehicle Networks

5.1 Introduction

Intrusion Detection Systems (IDSs) play a critical role in securing Controller Area Network (CAN). These systems broadly fall into two categories: rule-based and machine learning-based, each with its distinct strengths and limitations. Rule-based IDSs are cost-effective and operate by establishing rules derived from typical characteristics of normal CAN messages. However, these systems usually struggle with detecting sophisticated attacks. On the other hand, machine learning-based IDSs leverage advanced techniques like Neural Networks (NNs) to improve detection rates. Despite their relative effectiveness, they come with the burden of high computational costs and can be limited in detecting certain types of attacks.

To address these shortcomings, we propose a novel, flexible hybrid IDS framework that combines both rule-based and machine learning-based approaches. The IDS operates in two stages: a rule-based detection phase followed by a Deep Neural Network (DNN)-based detection phase. Rule-based detection in the first stage is used to catch attack messages that violate the established rules quickly. This first stage can reduce the workload of the second stage and further improve the efficiency of the hybrid IDS. DNN-based detection in the second stage is used to detect attack messages that fall out of the scope covered by the rules in the first stage. This design leverages the strengths of both approaches and compensates for their limitations, enhancing the system's

efficiency and effectiveness in detecting various types of attacks, which cannot be achieved by any individual detection method in this system. For example, rule-based detection cannot detect the masquerade attack, while machine learning-based detection is not able to identify the drop attack. Our experiments show that the proposed IDS achieves high accuracy and low latency compared with previous work.

Moreover, the hybrid IDS framework offers the advantage of adaptability, allowing for the replacement or alteration of the rule-based or machine learning components based on specific needs. Our proposed hybrid IDS framework offers an adaptable solution for securing the CAN network.

Contributions. This study makes the following contributions to the design of IDSs with higher detection accuracy and minimized computational cost:

- We propose a hybrid IDS framework consisting of two stages. The first stage incorporates robust and efficient rules to reduce the workload for the subsequently time-intensive second stage. In the second stage, advanced machine learning techniques are deployed to identify complex attacks that may bypass the first stage. This unique, hybrid approach allows for the detection of a broad range of attacks, something which each individual detection method could not accomplish independently. Furthermore, to avoid the imbalanced classification problem, our machine learning model is trained on datasets containing roughly equal proportions of malicious and benign messages.

- We consider a realistic adversary model. Unlike previous studies which often focus on specific attack scenarios, our model encompasses a broader range of known attacks in the literature, ensuring a more thorough examination of potential threats.

- Our proposed IDS is extensively evaluated using real traces collected from **four** different vehicles. This sets our work apart from previous studies that either rely on simulated data or data from no more than two vehicles. Given that actual IDs and data of CAN messages can vary across manufacturers, it is crucial to evaluate the IDS using data sourced from different

vehicle models.

Organization. This chapter is organized as follows. Section 5.2 presents the methodology of our design. Section 5.3 details insights into the implementation and evaluation. Section 5.4 provides further discussion of the proposed IDS. Section 5.5 concludes this chapter.

5.2 Our Design

Our objective is to build an IDS that achieves both accuracy and efficiency for securing the in-vehicle network. We propose a hybrid IDS consisting of two stages, a rule-based detection component, and a machine learning-based detection component, as illustrated in Figure 5.1.

The rule-based component is to offer robust and quick responses by catching malicious messages that clearly violate pre-defined rules. The machine learning component is to provide high accuracy by further detecting malicious messages which pass the rules in the first stage. This hybrid design helps to decrease the volume of inputs to the second stage. Moreover, this hybrid IDS can detect various types of attacks that cannot be detected by each individual detection method. In particular, we consider a comprehensive adversary model that covers almost all known attack types in the literature. Each detector in both components might be very efficient in handling certain types of attacks but totally ineffective for other types of attacks. For example, the valid ID rule can only detect messages with invalid IDs and has no way to detect the reply attack, the drop attack, and the masquerade attack. The time interval rule can only detect attacks that increase/decrease traffic volume but cannot detect the masquerade attack. The DNN-based component can handle most types of attacks, but cannot detect the drop attack. This proposed IDS with a hybrid design allows the whole system to achieve faster detection with high accuracy.

This IDS is supposed to be placed on a central gateway or installed on an ECU that can monitor the whole CAN bus traffic. As shown in Figure 5.1, offline training is conducted before the detection, and CAN traffic is the input for training. In the offline training process, rules are derived based on the features analyzed from the input, and the machine learning model is also trained for the

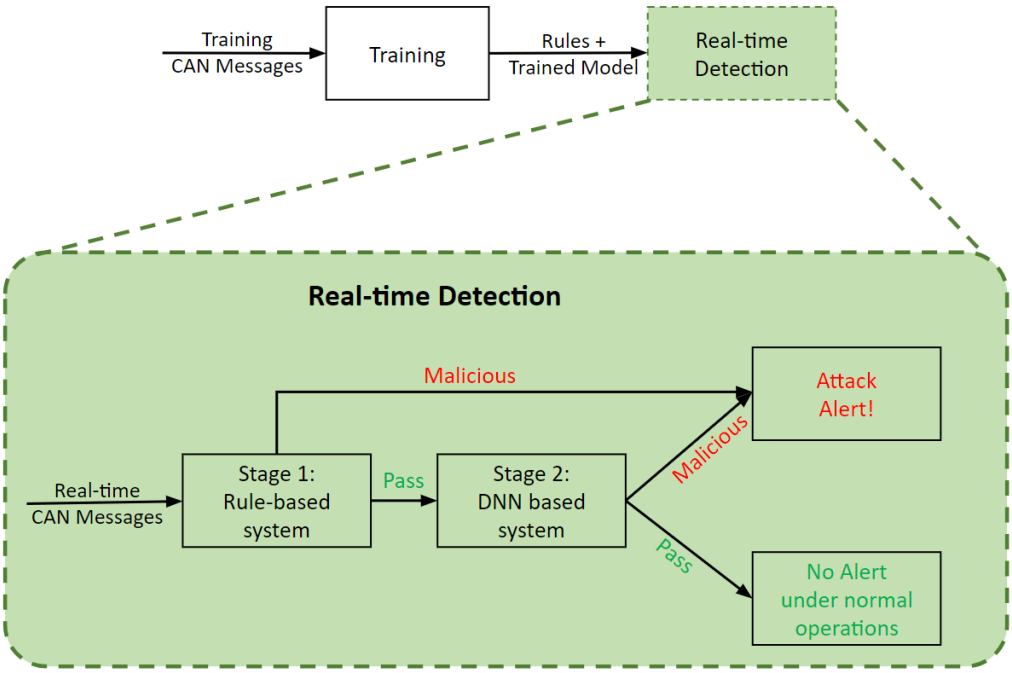

Figure 5.1: Workflow of the proposed two-stage IDS framework

detection task. When the IDS is applied, each CAN message transmitted on the bus goes through rule-based detection first. The CAN message that passes the first stage will be further processed by the second stage - machine learning-based detection. The message that passes both stages is considered a normal CAN message. Otherwise, the message is considered an abnormal message or a malicious message.

5.2.1 Stage 1: Rule-based System Design

Rule-based detection utilizes intrinsic patterns in the CAN data for anomaly detection. Rules are generally simple and straightforward. As a result, rule-based methods usually can offer fast detection. However, it might be difficult, or even impossible, to define a set of rules that can cover the normal behaviors of a complex system. Consequently, rule-based IDSs can have low detection rates and high false positive rates. In our IDS, instead of pursuing a high detection rate, the rule-based stage aims to provide fast detection and reduce the workload of Stage 2 without decreasing

the accuracy of the whole system. Therefore, we design Stage 1 that can do *fast detection* with *low false alarms*.

Various rules (such as time interval, CAN ID entropy, timestamp skew, CAN message ID sequence, etc.) have been proposed and used. Our challenge is to choose a set of rules based on our criteria, as not all the rules can meet these criteria. For example, the CAN message ID sequence rule [68] is based on the observation that a particular ID is followed by a specific set of IDs. It is effective in detecting all zero ID attack, or random ID attack, because those attack messages with invalid IDs violate the valid ID sequence directly. However, since it is hard to enumerate all possible sequence pairs, this rule can lead to a high false positive rate. Other rule-based IDSs also have their own weakness, as shown in Chapter 3 and Chapter 4.

For quick detection with low false positives, we select the following rules in our design:

Valid ID. Each CAN message comes with a valid ID that is specified in the DBC file. A message with an invalid ID is apparently abnormal. The valid ID list of a specific car model can be easily generated from a normal CAN trace or from the DBC file if it is accessible. When a message with an invalid ID is caught, it is thrown out of the system and will not be processed in the next stage.

Time interval. As shown in previous studies, most CAN messages are sent to the network periodically [14], thus, time intervals can be used as a rule. This rule works very well in detecting certain suspicious behaviors (such as injection attacks and drop attacks), which break the periodicity of periodic messages. To implement this rule, time interval values for each CAN message ID are needed. Those values can be generated from the normal CAN trace directly by calculating the difference between timestamps of two consecutive CAN messages with the same ID [101, 20].

Since this rule leverages the periodicity of CAN messages, it is exclusive to periodic CAN messages. We first analyze whether a message is periodic. We use Equation 4.1, 4.2 and 4.3 described in Section 4.2 to determine whether it is periodic.

For periodic messages, since their time interval values are distributed according to a normal distribution, $[\overline{T}$ - kσ, $\overline{T}$ + k$\sigma]$ is applied to find abnormal messages. Based on experiments, we set $k = 5$ as it leads lowest false positives. In real-time detection, T_i is calculated when a periodic

message arrives. If T_i falls in the normal range [$\overline{T}$ - 5σ, $\overline{T}$ + 5σ], the corresponding message is considered normal and will be moved to the second stage. Otherwise, the message is considered abnormal as it violates the time interval rule, and an alarm will be raised.

Valid DLC. DLC (data length code) is a part of CAN message that indicates the length of the data section of the message in bytes. The value range of DLC is from 0 to 8 inclusive. The actual DLC of CAN messages depends on the message design of a particular vehicle. Specifically, if CAN messages of a vehicle have different DLC values, the DLC of messages with the same ID is fixed. So when the DLC of a message does not match the valid value, the message will be caught and thrown out.

We follow a principle that rules that are simpler and make a quicker decision come first in order to achieve our goal. For example, "Valid ID" is the first rule we apply and it is the most straightforward one to quickly identify whether a message has a valid ID or not. In contrast, the "Time interval" rule is more complicated, needs more calculations, and thus is put at the last in the sequence. Concurrent execution of rules should work better but usually requires hardware support which is not considered in our design.

5.2.2 Stage 2: Deep Neural Network-based System Design

Machine learning has achieved remarkable performance on a variety of tasks including intrusion detection. However, it can be challenging to select a proper machine learning method for IDSs that can protect CAN. Our objective is to apply a machine learning method that can achieve high accuracy for intrusion detection. Deep Neural Networks (DNNs) are widely used in the field of intrusion detection [49, 50, 2, 23, 134]. As DNNs are constructed with multiple layers with multiple interconnected neurons, DNNs are able to model non-linear relationships and solve complex tasks. Previous studies have shown that DNNs outperform ANNs and traditional machine learning methods, such as support vector machine (SVM), random forest, and decision tree, in general, IDS tasks [121, 23, 134] or in CAN bus IDS tasks [49, 50]. We draw the same conclusion through our own experiments on comparing DNN with various traditional machine learning algorithms,

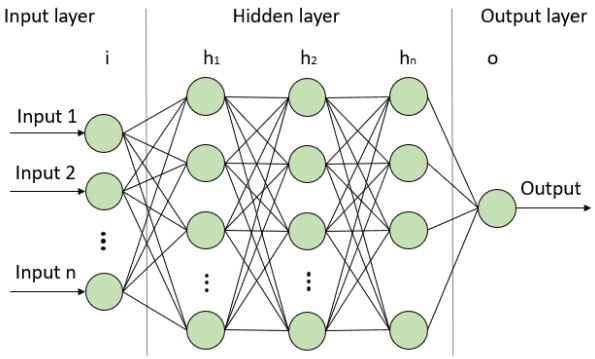

Figure 5.2: The DNN structure

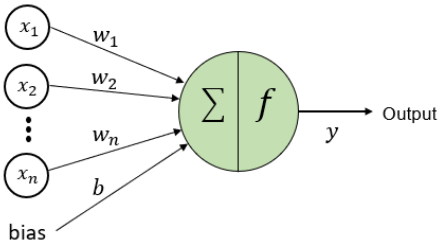

Figure 5.3: A single neuron

including Decision Tree, Random Forest, and SVM. Therefore, we select to use DNN in Stage 2.

A DNN consists of an input layer, multiple hidden layers, and an output layer. Each layer is composed of multiple neurons that are dots as shown in Figure 5.2. A neuron takes multiple features x_i as inputs, and each feature is associated with a weight w_i. The neuron calculates the weighted sum of inputs as the output, as shown in Figure 5.3. Usually, a bias b can be added as part of the calculation.

$$y = f(\sum_{i=1}^{n} x_i w_i + b) \tag{5.1}$$

Raw data can be used for training directly in deep learning [26]. This usually causes longer training time due to the processing of a large amount of data. Moreover, learning redundant or less important features can result in an overfitting model. To reduce the computational cost and prevent

overfitting, we choose to use features selected by feature engineering instead of using raw data. Based on the domain knowledge, features in consideration include those that can be extracted from raw data directly (e.g., ID, data field, timestamp, DLC) and the ones which can be derived from raw data (e.g., the number of occurrences for a certain message, ID sequence, the relative distance of ID entropy, changes in system ID entropy, the relative distance of data entropy, changes in system data entropy, occurrences of bit-symbol "1" or "0" in the CAN message data filed, the hamming distance between successive message data fields, the hamming distance between data fields of the successive message of the same ID). In addition, based on the comparison study in Chapter 3, we consider using both ID and data sections of CAN messages.

To select a good subset of features without losing too much information from the original data, we conduct the following steps. Firstly, we filter out features with low variance values [36], as a low variance value indicates the corresponding feature contains little knowledge. Secondly, we remove redundant features. That is, we calculate the correlation coefficient to find features that are highly correlated, then we only choose one feature as a representative for the next step. Thirdly, the random forest algorithm and stochastic gradient boosting algorithm are leveraged to calculate feature importance scores [11] among remaining features. Those algorithms calculate importance scores based on Gini and entropy. By employing those techniques on four datasets collected from four different vehicles, we finally select five features with high importance scores for training. It is noted that selected features can be vehicle model-dependent, because the actual messages IDs and data of a vehicle may vary from other vehicles from different manufacturers.

- **Message ID:** Message ID plays a crucial role in identifying malicious messages because each valid ID indicates a specific meaning by the manufacturer's design. That is, some messages with certain IDs should not be transmitted under certain vehicle operations. For example, CAN messages with diagnostic IDs should not be observed while driving, and CAN messages with invalid IDs should not be transmitted on the bus at any time. It can also serve as a link among messages with the same ID and assists DNN in finding hidden patterns.

- **Hamming distance:** The hamming distance between the data fields from two consecutive

messages with the same ID keeps a relatively steady value or falls in a small range. Therefore, abnormal behaviors can be detected by monitoring the variance of hamming distance [103]. We use D to represent the data field of a CAN message, $D = \{b^1, ... \ b^n\}$, where n can be up to 64-bit long and b^i is the i^{th} bit. Let D_t denote the data field of message t. The hamming distance between two consecutive messages (t and $t + 1$) is calculated as Equation 5.2:

$$H_d(D_t, D_{t+1}) = \sum_{i=1}^{n} b_t^i \oplus b_{t+1}^i, \tag{5.2}$$

where $\oplus$ is the exclusive or operation.

- **Entropy of data field:** Entropy values of data fields from normal CAN messages follow a relatively steady pattern, while entropy values from abnormal messages can break that pattern. Suppose there are m unique byte values in D (N-byte long): $B^1, \cdots, B^m$, where $m \leq N$, and $m = N$ when all the bytes in D have distinct values. The entropy of data filed D is calculated as Equation 5.3:

$$E(D) = -\sum_{k=1}^{m} p(B^k) log p(B^k), \tag{5.3}$$

where $p(B^k)$ is the probability of B^k occurrence, and log is the logarithm.

- **Bytes of importance:** Some bytes in the data fields contain more information than others. For example, certain bytes are never changed or may be reserved for future use. We set each byte as an individual candidate feature and calculate its importance score. And then, we select the top two bytes that have the highest feature importance scores. To note, the locations of the top two bytes are not fixed across all the vehicles because of the different designs of manufacturers. For example, the second and fifth bytes are considered the top two

for Ford Fusion 2013 dataset, but bytes in the same locations cannot be used as features for other vehicles.

We use feature vectors for feature embedding into this machine learning-based system. Each feature vector is a vector containing selected features about each CAN message. We put feature vectors to make feature space as inputs for the machine learning model. For each CAN message, selected features are processed to generate a feature vector $F = \{f_1, f_2, f_3, f_4, f_5, f_6\}$, where f_1 to f_6 are message ID, hamming distance, entropy of data field, bytes of importance 1 and 2, respectively.

5.3 Implementation and Evaluation

To evaluate the proposed IDS, we implement it on a Windows machine with Intel Core i5-4200U CPU @ 1.60GHz processor. We use Python version 3 for the implementation and TensorFlow as the library for machine learning.

The more hidden layers are used in a DNN model, the higher accuracy can be achieved at the cost of increased processing time [26, 49, 50, 2, 24]. A CAN IDS needs to be trained for each target vehicle model before detection because the actual CAN IDs and message semantics of a vehicle model are proprietary to each car manufacturer. Shallow neural networks generally rely on more carefully fine-tuned model structures and data processing. However, it can be challenging to fine-tune a specific shallow neural network model for each vehicle model every time. Our experiments which show that the performance is poorer when the number of hidden layers is less than three for each vehicle. Too many layers are also not preferred as they increase processing time. We also conduct experiments to evaluate models with different numbers of hidden layers. Results show that the computational cost increases as the number of layers increases, but the accuracy does not continue to increase, and sometimes it even leads to overfitting once the number of layers reaches a high enough value. For example, the detection rates with three and four hidden layers are 95.23% and 98.46% respectively, on Dataset 2. Detection rates with more layers (six to ten layers) are almost the same as the one with five layers. Through those experiments and also referring to

existing work [49], we select this DNN model with five hidden layers as a model that is generic enough to handle most vehicle types, at least, the four vehicle models used in our experiments. Specifically, we choose to use a feed-forward neural network model with five hidden layers with 100, 100, 80, 60, and 40 neurons at each layer, respectively. Only one neuron is in the output layer as the output is either 1 (positive) or 0 (negative) for the classification purpose. By following the typical hyperparameters and optimization strategies of neural networks for binary classification tasks [24], the ReLU activation function is used at each hidden layer while the sigmoid activation function is used at the output layer. The binary cross-entropy function is used as the loss function, and the Adam optimizer is used as the optimizer in our configuration.

We aim to make the proposed IDS works across different vehicle models, and our experiments show that it is effective on our four different datasets. However, for certain vehicle models, it may be necessary to modify the DNN structure, such as the number of layers or nodes, to achieve high accuracy. As an IDS framework, this IDS allows those future updates.

5.3.1 Experimental Datasets

As presented in Section 4.2, we collect data from the on-board diagnostics II (OBD-II) interface of real vehicles. The OBD-II port has been a mandatory requirement for all vehicles sold in the US. While its primary purpose is diagnostic and emission measurements, it also offers information, such as engine and body control. We collect CAN data by connecting the CAN Analyzer to the OBD-II port of a vehicle during regular operations.

To ensure a comprehensive evaluation of the proposed IDS framework, we carefully select the most representative datasets for evaluation. Four datasets are selected. Table 5.1 provides details of each dataset. The third row depicts the total number of messages in each dataset. The fourth row lists the number of unique message IDs which are dependent on manufacturers and vehicle models. The fifth row indicates the number of unique IDs considered periodical according to the periodic definition in Section 4.2. If the value of c_v calculated using Equation 4.3 is greater than 20%, the ID is considered as an aperiodic ID. Otherwise, it is considered a periodic ID. All messages in

Datasets 1, 2, and 4 are periodic messages, while 9 of 79 IDs in Dataset 3 are aperiodic.

Table 5.1: Experimental datasets

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Vehicle Model	Honda Accord 2006	Honda Civic 2018	Ford Fusion 2013	Chevrolet Volt 2013
Number of Messages	243,762	1,806,780	2,158,201	3,111,346
Number of Unique IDs	13	54	79	106
Number of Periodic Messages	All	All	70	All

5.3.2 Attack Strategy

We implement attacks mounted by a strong adversary, as depicted in Section 4.1 by using the following strategy:

- Injection Attacks:
  - *Random ID Attack (A0)*: inject 1 to 10 messages with random IDs.
  - *Zero ID Messages Attack (A1)*: inject 1 to 10 messages with the ID field set to zero.
  - *Replay Attack (A2)*: inject 1 to 10 consecutive previously-seen messages that are randomly selected.
  - *Spoofing Attack (A3)*: inject 1 to 10 messages which are generated based on the knowledge of CAN message specification. An example message is shown in Figure 5.4.
- Drop Attack (A4): drop a random number of normal messages in the range of 1 to 10.

- Masquerade Attack (A5): replace 1 to 10 normal messages with the same number of malicious messages that have the same ID and altered data content. This attack message has similar contents to the spoofing attack messages.

The medium adversary is able to execute A0, A1, A2, A3, and A4 while the weak adversary is only able to execute A0 and A1.

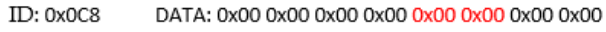

Figure 5.4: Example message for spoofing attack and masquerade attack

For A4 and A5, we need to understand CAN messages to generate corresponding malicious messages. Since we have no access to DBC files which are proprietary to car manufacturers, we do reverse engineering. For example, we find that CAN messages with ID 0x0C8 control the instrument cluster speedometer in Dataset 1. As shown in red in Figure 5.4, for ID 0x0C8, only the fifth and sixth bytes in the data section are changed under normal operations. By modifying the fifth and sixth bytes, the attacker is able to generate malicious messages to launch A4 and A5. For example, a spoofed CAN message can be generated with ID: 0x0C8 and Data: 0x00 0x00 0x00 0x00 0x01 0x02 0x00 0x00.

5.3.2.1 Training Datasets

When preparing the data to train a machine learning model for a classification task, we need to pay attention to the **class imbalance** problem. It is a problem in machine learning where the training dataset for each class label is not balanced, and it can result in the poor predictive performance of classification predictive models [37, 22]. To be an efficient IDS for in-vehicle networks, the IDS must accurately classify anomalies no matter whether the proportion of malicious messages to the total CAN traffic is large or small. Considering that intrusion detection is a real-world classification task where normal and abnormal message distribution is very likely imbalanced, we need to minimize the imbalanced problem to achieve better model prediction.

In order to achieve our goal and build a trained model that can satisfy the needs, we design a way to train the model with balanced data to avoid the imbalanced classification problem. First, we use the strong adversary model with randomized attacks to generate training datasets. The strong adversary is selected rather than others because it can randomly mount all types of attacks. Attacks (A0-A5) mentioned above are labeled from 0 to 5. Before a message is sent to the bus, a random number in $[0, 14]$ is selected. If the number is in the range of $[0, 5]$, the corresponding attack will be launched. Otherwise, a normal message will be sent. Therefore, each type of attack has a 6.67% chance of being generated for each message parsed. Second, since each message first needs to go through the rule-based stage, we let the rule-based component detect malicious messages, and only messages that can pass will be used to train the machine learning component. By this design, the number of malicious messages and the number of normal messages are comparable in the training dataset. As a result, the machine learning model can learn from a sufficiently representative number of examples of each class to classify attacks accurately.

5.3.2.2 Testing Datasets

Considering realistic in-vehicle cyberattacks where the number of attack messages may be small to avoid being detected, we use a different attack strategy in each test iteration where each attack has a 4% chance to be launched. For a strong adversary, we randomly choose a number in the range of $[0, 24]$ to decide whether an attack (A0-A5) needs to be launched or not before sending a message to the bus. For the medium adversary and the weak adversary testing datasets, A0-A4 and A0-A1 are randomly selected with a 4% chance, respectively, too.

5.3.3 Experimental Results

Effectiveness Evaluation:

To evaluate the effectiveness of the proposed IDS framework, we employ evaluation metrics that are widely used in IDS evaluation. Three metrics are used: accuracy, true positive rate (TPR, or detection rate), and false positive rate (FPR). Accuracy is the fraction of all correct predictions.

TPR is the fraction of detected messages that are truly malicious, and FPR is the fraction of detected frames that are not really malicious messages. Accuracy should be high as it indicates the overall detection ability, and TPR should be high as it shows the ability to detect malicious messages. FPR should be small as it means a low false alarm. Those metrics are calculated based on Equation 4.4, 4.5, and 4.8. By utilizing these metrics, we can provide a comprehensive assessment of our IDS, examining its ability to identify malicious activities while minimizing false alarms accurately.

It is noted that high detection rate (true positive rate) and low false alarm rate (false positive rate) are considered major performance criteria for IDSs generally. Especially, low false alarm rate is critical for in-vehicle networks [14], because the false alarms can cause unnecessary panic or waste resources, such as emergency services.

Table 5.2: IDS performance - strong adversary

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Accuracy	99.91%	99.81%	99.80%	99.85%
TPR	99.90%	99.75%	99.75%	99.81%
FPR	0.066%	0.071%	0.100%	0.084%
Time per msg	0.549 ms	0.551 ms	0.546 ms	0.553 ms

Individual Stage Result. To evaluate the effectiveness of the proposed two-stage IDS system, we first need to understand the effectiveness of each individual stage.

We experimentally evaluate the effectiveness of Stage 1 and Stage 2 separately under the same strong attack model. For the rule-based system, the average false positive rate is 0.005%, while the average true positive rate and the average accuracy are 83.25% and 88.83% respectively. The main reason for the low true positive rate and the accuracy is that masquerade attacks cannot be detected. For the DNN-based second stage, the average true positive rate is 81.67%, and the average accuracy is 87.72%. The cause of relatively low performance is that the machine learning-based component cannot detect the drop attack.

Two Stage Result. To analyze the effectiveness of our IDS, we use hold-out, in particular for

each dataset, we use 70% to build a training dataset for training and 30% to build a testing dataset for testing. Considering that a CAN IDS will receive CAN messages in a continuous manner and cannot access messages randomly from a traffic trace, we require training on the first part of a dataset and testing on the last part. Table 5.2 shows experimental results for the strong adversary. Each result is the mean value of 10 repeated experiments results as our adversary model randomly launches different types of attacks. For all datasets, our IDS achieves both accuracy and detection rate over 99%, and false positive rate lower than 0.1%. It can be seen that there is no clear relationship between the number of unique IDs and accuracy/detection rate. Accuracy and detection rate are not related to the number of periodic messages as well. Regarding the false positive rate, Dataset 3 has the highest false positive rate compared with other datasets. It might be because of aperiodic messages that cause more false positives.

Table 5.3: IDS performance - medium adversary

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Accuracy	99.94%	99.95%	99.95%	99.95%
TPR	99.92%	99.93%	99.93%	99.93%
FPR	0.027%	0.022%	0.024%	0.022%

Table 5.4: IDS performance - weak adversary

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Accuracy	99.99%	99.99%	99.98%	99.98%
TPR	99.99%	99.99%	99.99%	99.97%
FPR	0.014%	0.011%	0.018%	0.011%

Table 5.3 and 5.4 show how our proposed IDS performs under a medium adversary and a weak adversary respectively. From Table 5.2, 5.3 and 5.4, it can be seen that the performance against a medium or a weak adversary is slightly better than that against a strong adversary. Nevertheless,

the results are comparable in accuracy, detection rate, and false positive rate due to the use of a strong adversary model in training.

Experiment results above show the two-stage IDS performs better than individual stage as the two-stage helps to detect more types of attack (as mentioned in the purpose of each stage experiment).

Among existing machine learning-based approaches for CAN IDS, a DNN-based IDS [49] reports a 1.6% false positive error and 97.8% accuracy while an RNN-based IDS [126] reports a detection rate of 95%. One rule-based approach [14] reports a similar false positive rate of 0.055%. Compared with those results, our hybrid IDS provides promising detection rate, accuracy, and false positive rate. However, it is noted that the aforementioned schemes use different experimental platforms and adversary models.

Efficiency Evaluation: To provide efficiency analysis, we evaluate the time performance of the proposed IDS. As in-vehicle networks are time-critical systems that cannot accept long detection latency, we focus on the time delay during the detection process (we worry less about the training process as it can be completed offline.). As shown in Section 4.3, we estimate processing delay to evaluate the efficiency. The processing delay of each message is calculated in milliseconds (ms). The average processing delay per message is calculated over 10 experiments and shown in Table 5.2. Our IDS achieves an average processing delay in the range of [0.546, 0.553] ms, for all data sets, with a mean value of 0.550 and a standard deviation of 0.0026. It suggests that the processing time is neither related to the number of unique IDs nor affected by the number of aperiodic messages. Moreover, it is noted that the processing delay per message can be different under various attack models because the processing time depends on the number of malicious messages that are filtered out at the first stage. The average processing delay per message is 0.129 ms in Stage 1 and 0.885 ms in Stage 2. There are not many previous studies on machine learning IDS that report the processing delay. A processing delay of 2-5 ms is reported [49] while the testing platform information is not provided.

Performance on Different Attack Rates: To show that the proposed IDS can effectively detect

attacks regardless of attack rates, we also conduct experiments using test datasets with different attack rates: one contains a small fraction of malicious messages (1% chance for each type of attack), the other one contains a large number (8% chance for each type of attack) of malicious messages. The performance of our IDS under a strong adversary with a low attack rate is shown in Table 5.5, and the performance of the IDS under a strong adversary with a large attack rate is shown in Table 5.6. It can be seen that experimental results are similar to the performance shown in Table 5.2. These results prove that our IDS can effectively detect malicious messages regardless of whether the attack rate is high or low.

Table 5.5: IDS performance - spares attack traffic

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Accuracy	99.90%	99.89%	99.84%	99.92%
TPR	99.82%	99.75%	99.61%	99.90%
FPR	0.078%	0.062%	0.084%	0.076%

Table 5.6: IDS performance - dense attack traffic

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Accuracy	99.76%	99.87%	99.78%	99.86%
TPR	99.75%	99.86%	99.76%	99.86%
FPR	0.067%	0.066%	0.085%	0.071%

Performance on Different Amount of Training Data: Generally, less training data has low computation cost, but insufficient training data cannot provide a trained model that meets the accuracy requirement. On the other hand, too much training data can offer higher accuracy while it incurs long processing delays and needs large storage resources. To find the appropriate amount of training data, we conduct experiments with different amounts of training data. We analyze the different amounts of training data on the impact of performance for each dataset. Figure 5.5, 5.6, 5.7, and 5.8 show the experimental results. The horizon axis is the percentage of CAN messages for

training. The vertical axis represents the percentage of detection rate (in green) and false positive rate (in red). Those results indicate:

- The proper size of training data is different for each dataset. And results show a trend that the increasing of training data can decrease the false positive rate.
- Figure 5.5 indicates that more than 60%-70% data is needed to achieve the desired performance for Dataset 1, whereas about 20%-40% data is enough to offer desired results for the other three datasets. It is noted that it may vary on different vehicle models. Among all datasets, Dataset 1 and 3 have relatively lower detection rates when less than 50% of data is used for training. For Dataset 1, due to its small size, the small amount of training data should be the leading cause for the low detection rate. As for Dataset 3, the low detection rate may be caused by aperiodic messages. We may need more data for training when aperiodic messages are involved for acceptable results.

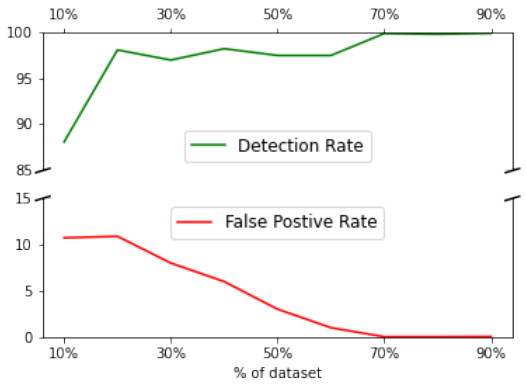

Figure 5.5: Performance over Dataset 1

Model Loss: In the context of machine learning, an epoch is a time period when an entire training data is passed through a machine learning model. As an important hyperparameter of a machine learning model, the number of epochs defines the number of passes the entire training data going through the machine learning algorithm. Usually, multiple epochs are needed to train a neural network. More epochs help the network to see the previous data and readjust the model

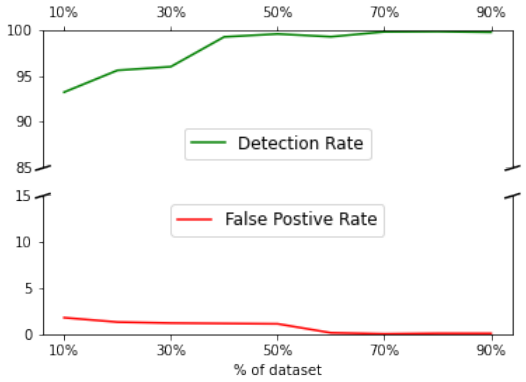

Figure 5.6: Performance over Dataset 2

Figure 5.7: Performance over Dataset 3

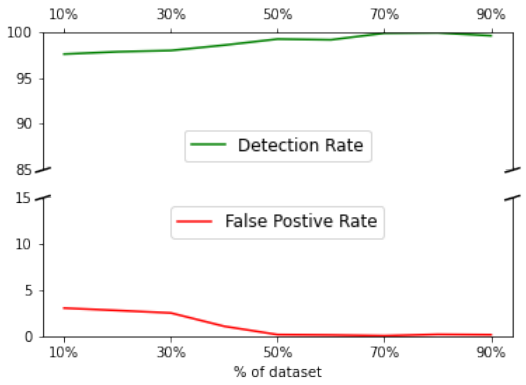

Figure 5.8: Performance over Dataset 4

parameters. However, too many epochs are not necessary and may bring the over-fitting issue. To find the proper number of epochs, we analyze the model loss for the machine learning component. In our experiment, we use the binary cross-entropy loss function and the Adam optimizer function that is under mini-batch gradient descent. We examine the number of epochs in the range 1 to 150 for each dataset and the results are shown in Figure 5.9. The horizontal axis is the number of epochs, and the vertical axis is the training error rate. In general, if the overall trend is going down, the fluctuations within certain limits are normal, which depends on the fact that a heuristic method is used as an optimization function. As shown in Figure 5.9, all models have converged, and each one has a reasonable loss. Models of Dataset 1, 2, and 4 have comparable performance while Dataset 3 has a higher loss at earlier epochs. Even though the difference is not big, it indicates that Dataset 3 needs more epochs for better performance. A possible reason is that Dataset 3 contains aperiodic messages. Generally, the training should be stopped when the error rate is minimum. We need to stop training at an earlier epoch when the value of loss starts to remain at a certain number consistently. Otherwise, it may generate an over-fitted model. Based on our experimental results, ten to twenty epochs are appropriate for Dataset 1, 2, and 4, while sixty to eighty epochs are needed for Dataset 3.

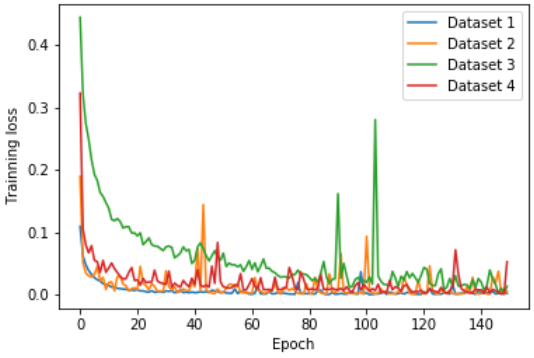

Figure 5.9: Model loss

Overfitting: We apply three approaches to prevent overfitting in this work. 1) hold-out. For each dataset, we use 70% for training and 30% for testing. In a real car, a CAN IDS will receive

CAN messages in a continuous manner, and it will not access messages at random from the traffic trace. Therefore, our IDS is trained on the first part of the trace and tested on the last part. Cross-validation is not applied because CAN IDSs cannot access messages at random from the traffic trace. 2) feature selection. When neural networks learn too many features, they can eventually overfit. Feature selection selects the important features for our model, which helps to prevent overfitting. 3) early stopping. We train our model for a large number of epochs, and we monitor the testing loss (using hold-out). If it begins to degrade, we stop the training and save the current model, which helps to prevent overfitting. By using those approaches, we do not find overfitting models in experiments on four datasets, but it may happen for CAN data of other vehicles. Since our hybrid IDS is a scalable framework, more approaches can be applied to prevent overfitting, such as L1/L2 regularization.

5.4 Discussion

As shown in Section 5.3, the proposed IDS can detect malicious messages effectively and efficiently. In this section, we list some remaining open questions, challenges, and limitations:

Root cause analysis: When a message is recognized as malicious, our IDS can respond by raising alarms. However, we do not provide a root cause analysis report for further troubleshooting. The root cause can be facilitated by a fingerprinting approach [14] that can be complimentary with our IDS. Our work focuses on intrusion detection, and root cause analysis is beyond the scope of this research.

Data collection: We collect CAN bus data through the OBD-II port. However, there are some practical limitations. Firstly, if a vehicle has multiple CAN buses, manufacturers may define some discretionary pins in the OBD-II port in addition to standard CAN high and CAN low pins. To collect data on all CAN buses, information about those pins is needed. Secondly, data collected directly from the OBD-II may be limited or hindered because of different CAN implementations by different vehicle manufacturers. Those potential issues can be solved by connecting to the CAN

bus directly, but it requires some levels of destruction for the vehicle.

Various driving conditions: We collect the datasets on the clear driving road with cautious driving (we only have one driver). Road conditions and driving styles might be very different in reality. Those may lead to some changes in the characteristics of CAN messages or even cause false alarms to our IDS. For example, driving on a slippery road or unique driving styles or habits of different drivers may cause changes in the CAN traffic. In our future work, we may collect data with multiple driving conditions to see how they impact the result.

Training module updates: During the life cycle of a vehicle, updates on the training model for detection may be needed. It is a problem for all machine learning-based IDSs. For example, the replacement or maintenance of some components may require a newly trained model. The aging of the vehicle is another concern that may require a refined trained model. The trained model can be generated offline by an authorized dealership. However, the updating process may expose the system to other threats or attacks.

Adversarial evasion: If an attacker can modify ECU(s) and cause a "modified" CAN trace for the offline training, the attacker can trick the IDS and lead to a "wrong" model. Then the malicious behaviors may bypass the detection of IDS. We make an assumption that the CAN trace collected for offline training is the CAN data without the adversary's manipulations.

Limitation: The proposed IDS is shown to be effective in detecting various types of in-vehicle network intrusions. One limitation is that it would be difficult to detect attacks that drop aperiodic messages. It is still considered an open problem for IDSs, especially timing-based schemes [7, 71, 76, 77, 101, 14, 102] that leverage the periodicity of CAN messages, to detect attacks on periodic messages. Despite the fact most CAN messages are periodic [14], and most intrusions are detectable, a solution is still needed to protect CAN from such attacks. Most existing studies do not cover this type of attack as they generally focus on the injection attack [68, 103, 114, 102, 50, 49, 2, 7, 71, 76, 77, 101, 14, 102]. Possible solutions would be leveraging detection features rather than time, such as the CAN messages sequence, and cyber-physical features.

5.5 Conclusion

This chapter proposes a hybrid IDS model to secure the in-vehicle CAN bus and to protect vehicles from malicious intrusions. The proposed IDS combines traditional rule-based IDS techniques with emerging machine learning methods to achieve a balance between efficiency and detection accuracy. Datasets collected from four vehicles of different models and manufacturers have been used to evaluate the effectiveness and efficiency of the proposed scheme. Our experiments show that this hybrid IDS can detect more types of attacks compared to each individual detection method and can also improve efficiency. The promising results demonstrate that the IDS is able to detect malicious behaviors and can work for different vehicle models. Our study can serve as a proof of concept that our hybrid IDS model has the potential to enhance in-vehicle security significantly.

CHAPTER 6

Accelerating In-Vehicle Network Intrusion Detection System Using Binarized Neural Network

6.1 Introduction

In Chapter 5, a hybrid IDS framework is introduced, which utilizes both rule-based and machine learning-based approaches to achieve effective and efficient intrusion detection for CAN. While this system already delivers high accuracy and satisfactory computational efficiency, the challenging real-time operation of in-vehicle systems requires even better performance. Particularly, our deep learning component, despite its robust accuracy, can be more efficient. In this chapter, we focus on improving this component, aiming to enhance its computational efficiency while maintaining satisfying accuracy significantly.

Machine learning approaches have been deployed to develop IDSs due to their remarkable capabilities in complex applications [59, 45, 102]. However, they generally suffer from high computational complexity, usually require considerable computing resources, and often result in large power consumption. A neural network model can require hundreds of Megabytes (MBs) of memory [100, 140, 6, 39]. This requirement for memory access is often the bottleneck of system performance as well as energy efficiency [104]. Therefore, deploying neural networks on resource-constrained devices in embedded environments can be difficult. Considering the constraints of ECUs' memory and the demand for real-time communications of in-vehicle networks, an IDS for in-vehicle networks should be small in size and have a fast response time during the detection

process.

To overcome those limitations and protect CAN against vehicle cyberattacks, we propose a new IDS based on Binarized Neural Network (BNN) that uses binary values for activations and weights instead of full-precision values. The BNN model can accelerate intrusion detection and reduce memory requests as well as energy consumption compared with the corresponding full-precision neural networks.

Our BNN model employs an input generator to convert CAN data into a grid format, enhancing its capacity to learn temporal sequential patterns, without the need for additional data preprocessing. Beyond the inherent acceleration of BNNs, our system can utilize Field Programmable Gate Arrays (FPGAs) for further speed enhancements. FPGAs, with their low power consumption and high flexibility, are ideal for embedded systems. Yet, their application in complex neural network models has been limited due to constraints on computing and memory resources. The bitwise nature of BNNs, however, makes them compatible with FPGA acceleration, offering a potential solution to these limitations [1, 120, 61, 75, 141, 91, 31, 116, 95].

BNN models are compact and efficient due to binarized weights and activations. However, accuracy is usually sacrificed to some degree. Improvement strategies can be applied to mitigate the loss of accuracy [64, 12]. As our goal is to utilize BNNs with higher accuracy while maintaining relatively low latency and memory cost, we explore various design choices of BNNs, including increasing network width and depth methods, to investigate if and how those methods can improve accuracy.

To evaluate the proposed scheme, we use datasets from four real vehicles from different car manufacturers. Our experimental results show that the proposed IDS can achieve satisfying detection rates as well as low latency. We also test our IDS on FPGAs and other platforms to show the performance of accelerated inferences, and we evaluate the memory utilization and power consumption on different platforms, including CPUs, GPUs, and FPGAs. Moreover, we investigate BNNs with different design choices to see how they can improve accuracy performance. Our experimental results demonstrate that wider and deeper models provide better accuracy performance at the cost

of increased latency and model sizes to varying degrees.

Contributions. This study makes the following contributions:

- To the best of our knowledge, this is the first work applying BNN to IDS for in-vehicle networks. We adopt the BNN model [44, 8] and convert CAN bit-stream traffic to the sequential pattern of CAN messages to feed into our BNN model.
- Our BNN-based IDS has lower latency (3 times faster) compared with another full-precision NN-based IDS. It also requests a smaller memory size as well as lower power consumption. Small model sizes and low power consumption are important for in-vehicle network environments as they consist of resource-constrained embedded devices. Moreover, the proposed BNN-based IDS can be further accelerated by executing on FPGA hardware (128 faster than on an embedded CPU).
- We evaluate and analyze various BNN models to determine how the width and depth of a BNN can affect performance. Our experiments show that both deeper and wider models can offer higher accuracy. Different applications can choose different models according to their specific needs and available resources.

Organization. This chapter is organized as follows. Section 6.2 presents the design of the proposed IDS. Section 6.3 delves into the details of the scheme's implementation and evaluation. Section 6.4 discusses the proposed IDS and related open questions. Finally, Section 6.5 concludes this chapter.

6.2 Our Design

Our objective is to build an in-vehicle network IDS to protect the CAN bus by detecting suspicious or malicious activities in the bus. Meanwhile, this IDS aims to have low latency on the detection process. It is supposed to be placed on a central gateway or installed on an ECU as a node of the CAN bus that can monitor the CAN traffic.

We propose a new IDS using BNN to identify intrusions in order to overcome limitations on existing machine learning-based IDSs and protect in-vehicle networks against vehicle attacks. BNNs are types of neural networks using one-bit quantization for weights and activations, and they can provide shorter inference time and smaller memory requests than corresponding full-precision neural networks. The actual IDs and data of a vehicle's CAN messages depend on the manufacturer's design, and this information (usually defined in the manufacturer's proprietary DBC file) is kept strictly and not published for security reasons. We use raw CAN messages that are transmitted on the CAN bus and leverage the sequential patterns of messages to detect malicious intrusions without the need for DBC files.

As illustrated in Figure 6.1, the proposed BNN-based IDS consists of two steps: 1) the training step and 2) the detection step. In the training step, we use labeled inputs for training. It can be challenging for the BNN model to directly learn CAN messages. Therefore, we design an input generator that assembles ten consecutive CAN messages and attaches corresponding labels to build input frames. If one or more attack messages are in one input frame, the frame label is set as malicious. Otherwise, the label is set as normal. In this way, BNN is able to learn sequential patterns during the training process. Note that training can be performed offline as training a classifier is considered time-consuming. Once the training process is completed, the trained BNN model is used in the detection process. In the detection step, the input generator only assembles current traffic messages as inputs without attaching labels. Those inputs are processed by the trained BNN model, where the topology and parameters are learned from the training step. The model can be deployed in general-purpose hardware such as CPUs, GPUs, or FPGAs. The output of the model is a prediction of whether the input is malicious or not. If an input is identified as malicious, an alarm will be raised.

6.2.1 Input Generator

Neural networks are able to take raw data as inputs without any extra hand-designed features. Our goal is to process raw CAN traffic data without feature engineering efforts that also save the

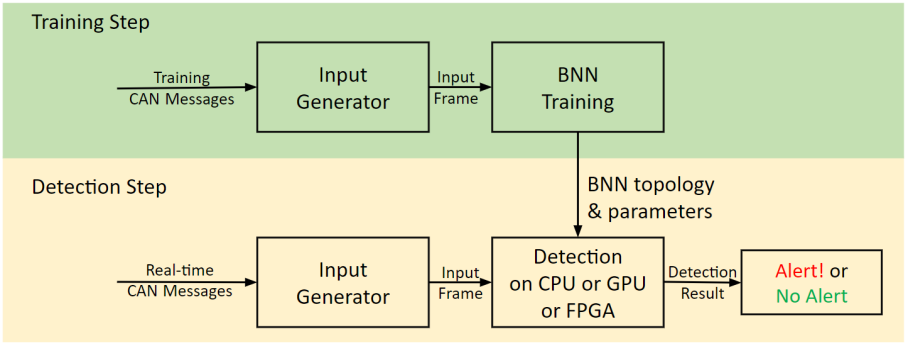

Figure 6.1: Workflow of the proposed BNN-based IDS

corresponding processing time.

As described in Section 4.1, an attacker can launch various attack types by injecting different types of malicious messages. Figure 6.2 provides two CAN traffic log segments without and with attacks. Each line shows one message. Both "normal" and "attacks" segments are collected for 7 ms. In normal conditions, the number of messages with ID 064 occurrences is 2. However, when there is an attack, the number of message occurrences increases to 9 messages. As a result, the sequential pattern of the CAN traffic is changed, which can be exploited to detect attacks.

```
ID: 12C   Data: 7B 72 00 00 02 EB 11 CC
ID: 064   Data: FF C0 00 00 EE 00 8A CC
ID: 188   Data: C6 06 00 02 00 00 0E CC
ID: 090   Data: 02 04 FF C1 00 0F 08 DB
ID: 1C0   Data: 00 00 00 00 00 00 00 00
ID: 0D4   Data: 00 E0 00 48 56 00 00 CF
ID: 0A6   Data: 00 00 09 0A 00 00 00 0D
ID: 0C8   Data: 00 00 00 3B 00 57 00 06
ID: 12C   Data: 7B 72 00 00 02 E6 11 0D
ID: 188   Data: C6 06 00 02 00 00 0E 08
ID: 20C   Data: 02 54 00 00 02 E6 00 87
ID: 0D4   Data: 00 E0 00 48 56 00 00 0B
ID: 1F4   Data: 04 64 00 01 3D F7 00 80
ID: 0A6   Data: 00 00 09 0A 00 00 00 49
ID: 0C8   Data: 00 00 00 3C 00 57 00 41
ID: 064   Data: FF C0 00 00 EE 00 8A 08
ID: 12C   Data: 7B 72 00 00 02 E6 11 49
ID: 090   Data: 02 05 FF C1 00 0F 03 1B
```

```
ID: 0C8   Data: 00 00 01 7E 01 90 00 C4
ID: 064   Data: FF C0 00 00 EE 00 65 4B
ID: 12C   Data: 78 71 00 00 02 EA 10 C2
ID: 090   Data: 02 05 00 27 00 0F 0F 5D
ID: 188   Data: 21 01 80 08 00 00 0E C2
ID: 1C0   Data: 03 1C 06 50 0C 20 19 67
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 0D4   Data: 00 E0 00 45 65 00 65 C7
ID: 0A6   Data: 00 00 09 09 00 00 00 0E
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 0C8   Data: 00 00 01 81 01 94 00 08
ID: 12C   Data: 78 71 00 00 02 EA 10 0E
ID: 188   Data: 21 01 80 08 00 00 0E 0E
ID: 20C   Data: 02 54 00 00 02 EA 00 0B
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 0D4   Data: 00 E0 00 45 65 00 65 03
ID: 1F4   Data: 04 64 00 01 3D F7 00 C0
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 2E4   Data: 57 00 00 00 00 00 00 8C
ID: 0A6   Data: 00 00 09 09 00 00 00 4A
ID: 0C8   Data: 00 00 01 83 01 97 00 4F
ID: 064   Data: FF C0 00 00 EE 00 65 87
```

Figure 6.2: CAN traffic log without attack (left) and with attack (right)

To enable the BNN model to exploit the sequential pattern of CAN traffic, we design an input

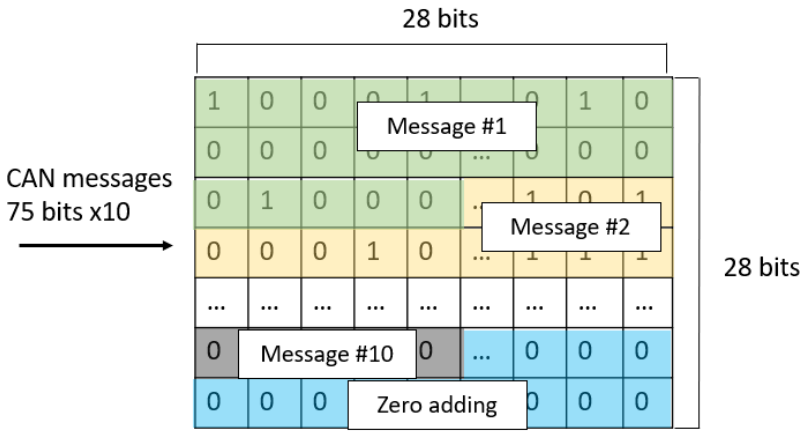

Figure 6.3: An example input frame generated by the input generator

generator that transforms consecutive CAN messages to grid structure frames, and then the BNN model learns the sequential pattern of these input frames. Each input frame contains data bits extracted from N consecutive messages and padding bits. In our design, the generator extracts 11 bits from the ID section and 64 bits from the data section, and the bit representation of each message is as follows:

$$M = b_0^{id} \cdots b_{10}^{id} b_0^{data} \cdots b_{63}^{data} \tag{6.1}$$

We conduct experiments to find the proper size N of the input frames and report details in Section 6.3.4. Based on experiments, we set $N = 10$, i.e., the input generator assembles every ten sequential CAN messages into an input frame. Figure 6.3 shows an example input frame. In the last ten consecutive CAN messages, the generator extracts 75 bits from each message (11-bit ID section and 64-bit data section) for a total of 750 bits from ten messages, plus 34 bits for padding. Then, the generator builds a 28 × 28 bitwise frame by stacking these 784 bits. By utilizing this input generator, no hand-designed features are needed, and our IDS can also be more efficient through batch processing compared to other IDSs [136, 137], because multiple CAN messages in a frame

are processed at once during detection.

Many CAN IDSs [102, 101, 68] only rely on the ID field of CAN messages for intrusion detection. Therefore, they are ineffective to detect malicious messages that contain valid ID and malicious data content, such as spoofed messages. Unlike those schemes, our IDS learns from input frames that include full information about each message and is able to detect such attacks.

Using the signal data (0 or 1), rather than decoded values, does not introduce extra prepossessing. Considering that thousands of messages are transmitted per second on a CAN bus, such computational efficiency should be taken into account for an IDS design. Furthermore, neural networks generally need to normalize data to improve accuracy, because a dominating feature with a bigger scale compared with other features causes low accuracy. Our model does not need normalization for input data, which also further improves efficiency by avoiding the computation cost on normalization, as all inputs consist of only 0 or 1. In addition, the 0 or 1 data helps our IDS map to FPGA hardware more efficiently because these data make bit manipulation easier for the hardware.

6.2.2 Binarized Neural Network

Unlike common neural networks which calculate the parameters in floating-point format, BNNs use binarized weights and activations instead of full-precision ones. By using bit-wise operations, BNNs can substantially reduce computation costs and improve efficiency. In full-precision neural networks, the basic operations usually can be expressed as Equation 6.2, where z is the output tensor and σ represents a non-linear function. w and a represent the weight tensor and activation tensor, and their outer product is decoded as $\otimes$.

$$z = \sigma(w \otimes a) \tag{6.2}$$

In BNN, during the forward propagation stage, floating-point weights w and activations a are replaced by w_b and a_b that are all 1-bit long, which can be defined as follows:

$$Q_w(w) = \alpha w_b, Q_a(a) = \beta a_b \tag{6.3}$$

A BNN can be reformulated in the regular DNN format as follows:

$$z = \sigma(Q_w(w) \otimes Q_a(a)) = \sigma(\alpha\beta(w_b \odot a_b)), \tag{6.4}$$

where $\odot$ is an inner product for vectors that operate using XNOR-Bitcount [93].

6.2.2.1 Binarization in BNN

The essential idea of BNN is to set weights and activations to +1 or -1 [44, 8]. Two binarization methods have been introduced: stochastic or deterministic. We use the deterministic method as it does not require hardware to generate random bits during quantization. Since the output from our input generator is either 0 or 1 without any normalization process, the first input of BNN and the rest of the layers can be both formulated as Equation 6.5:

$$x^b = Sign(x) = \begin{cases} 1 & \text{for} \quad x \geq 1 \\ -1 & \text{for} \quad otherwise \end{cases} \tag{6.5}$$

The problem is that the derivative of the sign function gets almost zero everywhere while in the backpropagation process. [44] uses Straight-Through Estimator (STE) in a deterministic way, which is called hard tanh ($HTanh$):

$$HTanh(x) = \begin{cases} 1 & \text{for} \quad x \geq 1 \\ x & \text{for} \quad -1 < x < 1 \\ -1 & \text{for} \quad x \leq -1 \end{cases} \tag{6.6}$$

By using STE, the 32-bit floating-point weights, denoted as W_{32bit}, are updated with an optimization strategy. During a large portion of training, a positive value of W_{32bit} is evaluated to have a positive gradient, and that value is increased in every update. If values in W_{32bit} are not bounded,

they will be accumulated to large numbers. For this reason, BNNs clip the values of W_{32bit} between -1 and +1. This keeps the values of W_{32bit} closer to binarized weight W_{1bit}.

6.2.2.2 BNN Model Design

We design a simple but powerful fully-connected network. As shown in Figure 6.4, this network takes input frames (each frame consists of ten consecutive CAN messages) from the input generator and outputs a binary decision indicating whether or not there are any malicious messages among the ten messages. We set three hidden layers for the network and 1024 nodes for each hidden layer as a baseline BNN model. In each hidden layer, there are three sub-layers: 1) a binarized dense layer. It is deeply connected with its preceding layer. All weights and activations are binary values, except for the first binarized dense layer where only weights are binarized; 2) a batch normalization layer. It standardizes the inputs to a layer for each mini-batch, which helps prevent overfitting; 3) a dropout layer with a 15% dropout rate. It is to reduce overfitting and improve the generalization of deep neural networks.

We adopt training strategies described in [17], including shift-based batch normalizing and AdaMax. To train our model, and make it more efficient to be mapped to FPGAs without affecting the network accuracy, additional optimizations are applied:

- Converting the input frames to 0 or 1 through the input generator makes it easier for hardware to process the bitwise operations. Then 1-bit values are used for all input activations, weights, and output activations to make the model full binarization, where an unset bit represents -1, and a set bit represents +1.

- Based on [116], the summation of a binary dot product can be implemented on hardware by a popcount operation where the set bits are counted by avoiding signed arithmetic accumulation.

In the detection step, almost all parameters in the BNN are binarized during the inference stage. There are 3.95 million multiply-accumulate operations, but 3.12 million of them are binary multiply-accumulate, which makes the network work fast if it is deployed on custom BNN hardware.

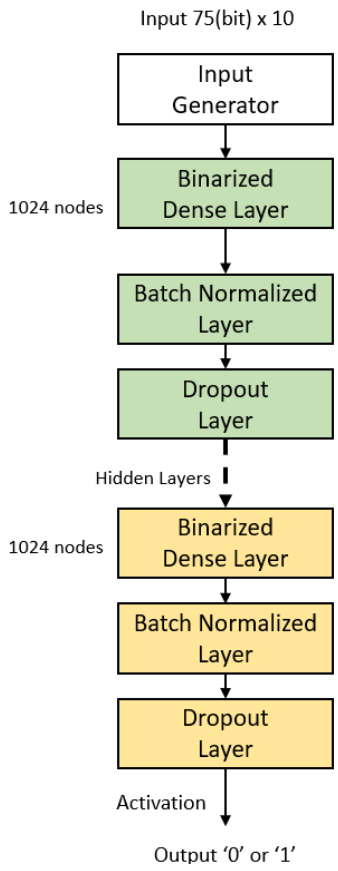

Figure 6.4: Baseline BNN structure

6.2.2.3 BNN Inference on FPGAs

While a BNN model is trained on powerful computational devices, mostly on CPUs or GPUs, the trained model is able to be deployed on general hardware. Previous studies [31, 116, 95] show that the BNN can work on resource-constrained devices, such as embedded CPUs, ASICs or FPGAs. FPGAs are considered as one of the most widely used platforms for BNNs. FPGAs offer hardware customization and can be programmed to deliver performance similar to GPUs or ASICs. The reconfigurable nature of FPGAs lends itself well to the rapidly evolving AI landscape and marketing requirements. In this work, we utilize an FPGA inference framework from Xilinx [8], which is originally designed for computer vision tasks, such as image classification. Because the input generator transforms the CAN traffic to input frames, we make our BNN-based IDS also work on this framework while enjoying further acceleration from the FPGA during the inference. Figure 6.5

illustrates the general workflow of converting a trained IDS into an FPGA accelerator. IDS supplies frames and a trained BNN to the Finn synthesizer. The synthesizer first determines the resource allocation by Matrix–Vector–Threshold Unit (MVTU), a computational core for the accelerator hardware design, to meet the frames per second (FPS) target and applies the optimizations. And then, it produces a synthesizable network description of a heterogeneous streaming architecture [116], an FPGA parallel optimizing design. In the last step, with the hardware library and Vivado HLx design suite, the data stream can be processed on the target FPGA platform.

Although the size of a full-precision neural network may be practically implemented on some advanced FPGA hardware, a smaller model size is always beneficial for CAN IDSs, given the constraints of ECU memory, the need for real-time communication of in-vehicle networks and budget-friendly considerations [35] (advanced FPGAs can be high-cost). Our BNN-based IDS takes advantage of neural networks and can accelerate CAN intrusion detection with a lower power consumption, a smaller model size, as well as lower budgets.

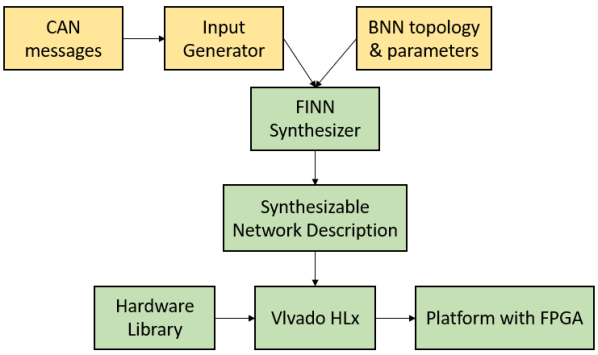

Figure 6.5: FPGA inference workflow

6.2.3 Energy Consumption and Memory Cost

Enhancing the computational performance for various tasks in neural networks remains an intricate challenge. In the last decade, power consumption is one of the dominant constraints on performance. This challenge has led to many research studies exploring ways to reduce energy usage in neural

networks.

Due to its unique properties, the BNN has been identified as a potential solution to these energy efficiency challenges. BNNs primarily offer power-saving benefits due to two main reasons: First, the binarization of weights and activations in BNNs allows for replacing a significant number of 32-bit arithmetic operations with more power-efficient bit-wise operations. This replacement drastically reduces the system resources required for computation, leading to increased power efficiency. Second, BNNs tackle the issue of energy-intensive data movement within the memory, which can often exceed the energy costs of computation. As per the findings reported in [133], in neural networks like GoogLeNet, only 10% of total energy is consumed due to computation, whereas 68% of energy expenditure is due to moving the feature maps. This disproportion underscores the significant power-saving potential of BNNs.

Compared to full-precision DNNs, BNN-based IDS require less memory space, making it possible to fit within on-chip memory. This attribute substantially reduces power wastage associated with memory access. Thus, by adopting BNNs with their efficient operations and reduced memory requirements, the goal of energy-efficient neural networks becomes more achievable. These advancements suggest a promising future in overcoming power consumption challenges and pushing the limits of neural network performance.

6.3 Implementation and Evaluation

6.3.1 Experimental Datasets

Similar to Chapter 5, where we utilize representative datasets, we employ the same datasets to evaluate the proposed IDS. By using these datasets, we enable consistent, direct comparison of experimental outcomes across different schemes proposed in this dissertation. Table 6.1 presents detailed information regarding each dataset used in the evaluation. The third row of the table displays the total number of messages contained within each dataset. The fourth row provides the count of unique message IDs, considering variations based on manufacturers and vehicle models.

Table 6.1: Experimental datasets

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Vehicle Model	Honda Accord 2006	Honda Civic 2018	Ford Fusion 2013	Chevrolet Volt 2013
Number of Messages	243,762	1,806,780	2,158,201	4,536,342
Number of Unique IDs	13	54	79	106

6.3.2 Attack Strategy

Section 4.1 discusses the different types of attacks that potential adversaries might launch based on their capabilities. Given that the proposed BNN-based IDS is a machine learning-based approach, the drop attack cannot usually be detected. However, considering this IDS can be incorporated within the two-stage framework (which has the capacity to address drop attacks, as discussed in Chapter 5), the BNN can be employed as a component of the second stage. In this work, we consider that attackers can mount attacks by injecting malicious CAN messages into the bus via an extra ECU or a compromised ECU.

Considering realistic attacks in the real world and existing studies on attack strategies [71, 73, 72, 14, 102], we implement 100, 200, 200, 300 intrusions on Dataset 1, 2, 3 and 4 respectively depending on their size. Each intrusion is performed for 3 to 4 seconds. For the *all zero ID* attack, we inject a CAN message with 11 zero bits ID every 0.3 ms. The purpose of this attack is to make the network resource unavailable to intended normal functions. Since all zero ID is the most dominant CAN ID in the bus, zero ID messages always win the bus in the arbitration phase, and the DoS effect can be achieved. Consequently, all other normal CAN messages cannot be transmitted. For the *random ID* attack, we follow a similar way to the all zero ID attack with different content of malicious messages. ID and data of those malicious messages are randomized. This type of attack can result in malfunctions of the vehicle by disturbing certain functions. For both *replay*

and *spoofing* attacks, we inject malicious messages every 0.8 ms. Malicious messages of the replay attack are valid messages transmitted previously on the bus. And malicious messages of the spoofing attack are altered messages based on valid messages with valid IDs and modified data. Understanding the format of CAN messages is needed for generating those malicious messages, which requires a process of reverse engineering. For instance, in Dataset 1, we find that the instrument cluster speedometer is controlled by CAN messages with ID 0x0C8. Under normal operations, only the fifth and sixth bytes in the data section change. By modifying these bytes, malicious messages can execute a spoofing attack. For example, a spoofed CAN message CAN be constructed with ID: 0x0C8 and Data: 0x00 0x00 0x00 0x00 0x01 0x02 0x00 0x00. For the hybrid attack, we insert a malicious message randomly chosen from all four types of attacks every 0.3 ms.

If the attacker's goal is to intentionally control the vehicle, the attacker should design malicious messages with specific IDs and data to the bus. Generally, all malicious messages and normal messages can be listened to by the target victim ECU. In order to ensure that malicious messages, instead of normal messages, can be received, the attacker needs to send malicious messages at a higher frequency than normal messages. In a typical message injection attack, the number of attack messages should be twice the number of normal messages. In practice, the number of attack messages can be significant (20-100 times) higher than the normal messages [72].

6.3.3 Experimental Setup

We design and conduct experiments on different platforms. To evaluate the effectiveness and efficiency of the BNN-based IDS, we use a regular desktop CPU to demonstrate the performance of our IDS in terms of accuracy and inference time on software. We use Python version 3 for the implementation and Theano/TensorFlow as libraries for machine learning. The BNN training device is a Linux machine running Ubuntu 20.04.2 LTS with a GPU device. For the hardware-based acceleration evaluation, we focus on embedded computing which aims to simulate a more realistic environment. We use two different platforms, including an embedded CPU and an embedded FPGA, which are low-power devices. Additionally, we use desktop CPU/GPU devices for comparison,

where we compare IDS detection latency and power usage effectiveness among those devices. Details of different platforms are listed as follows:

- Embedded FPGA: Xilinx PYNQ family Artix-7 FPGA, which has 13,300 logic slices, each with four 6-input LUTs and 8 flip-flops
- Embedded CPU: Arm Cortex-A9 processor with dual-core and 650MHz clock
- Desktop CPU: AMD Ryzen 3700x with 8 cores and 16 threads, base clock at 3.6GHz
- Desktop GPU: Nvidia RTX 2070 Super, 8GB DDR6 RAM

6.3.4 Input Generator

Our study introduces an input generator that compiles numerous successive CAN messages into composite input frames. Each constructed frame contains '*N*' individual messages and padding bits. The primary function of this structure is to enable the BNN model to exploit the sequential patterns in the CAN traffic.

We perform experimentation to demonstrate the input generator's critical role and determine the optimal size '*N*' for the input frame. Our tests show that when IDS is used without an input generator, feeding CAN messages directly into a BNN model, its performance can be significantly worse. In fact, without the input generator, its accuracy is only 78.43% against the hybrid attack on Dataset 1.

To determine the ideal frame size '*N*', we examine IDS performance for different frame sizes, including 5, 10, 20, and 50. Results show that when '*N*' is small, these frames may not adequately represent sequential patterns, affecting IDS performance. Conversely, larger input frames improve performance because they contain more messages and thus provide more representations of sequential patterns. However, our tests show that this improvement is not linear: performance does not consistently increase as frame size increases. After a certain point, increasing the frame size does not yield better performance.

Considering these experimental results, we design the input generator to generate frames encapsulating ten consecutive CAN messages, achieving a balance between providing sufficient sequential context and maintaining computational efficiency. This approach ensures that the BNN model can effectively exploit the sequential patterns of CAN traffic for intrusion detection.

6.3.5 Overfitting Prevention

In order to prevent overfitting, we apply the following approaches.

1. hold-out. For each dataset, we use 70% for training and 30% for testing. Considering that a CAN IDS receives CAN messages in a continuous manner and does not randomly access messages from the traffic trace, the first part of CAN traffic data is used to train our IDS, while the last part data is used to test. It is noted that no cross-validation is applied because CAN IDSs cannot access messages at random from the traffic.

2. regularization. Using regularization during training can mitigate overfitting. Regularization consists of various methods. We use two methods: batch normalization and dropout layer. Batch normalization helps to prevent overfitting by reducing the internal covariate shift and instability in distributions of layer activations. Moreover, it also results in the acceleration of the model optimization and better overall performance. Dropout layers prevent overfitting by randomly dropping a predefined ratio of nodes in the network.

3. characteristics of BNNs. Adding noise to activations and weights during training is one of the methods to prevent overfitting. In BNNs, both the activations and the weights are binarized, which can be considered a variant of the dropout method [17]. It helps to better generalize the model and reduce overfitting.

4. early stopping. We train our model for a large number of epochs, and we measure how well each iteration of the model performs. We stop the training and save the current model when it starts to degrade, which helps to prevent overfitting.

By using those approaches, we do not find overfitted models in experiments on four datasets, but it may happen for CAN data of other vehicles. As this BNN-based IDS is a scalable framework, more approaches can be applied to prevent overfitting, such as feature selection. If neural networks learn too many features, they can eventually overfit. Feature selection helps to select important features to reduce overfitting.

6.3.6 Experimental Results

6.3.6.1 Experiment 1: BNN vs. Full-Precision NN on CPU

This study aims to contrast the inference time between BNNs and full-precision 32-bit Neural Networks (NNs) that share an identical structural configuration. All of these computational tests were conducted on a uniform CPU platform to ensure a consistent baseline, guaranteeing that any observed differences in performance are attributable solely to the respective characteristics of BNNs and 32-bit NNS, not any variations in computational power or platform specifications. To effectively assess the robustness of these models under distinct types of attacks, we take into consideration a broad range of existing research studies alongside some of the most innovative and cutting-edge work within the realm of CAN IDS [14, 122, 136]. We use the same metrics in Chapter 5: accuracy, true positive rate (TPR, or detection rate), and false positive rate (FPR). Accuracy is the fraction of all correct detection results. TPR is the fraction of detected messages that are truly malicious, and FPR is the fraction of detected messages that are not really malicious. We utlize Equation 4.4, 4.5, and 4.8 defined in Section 4.3 to calculate them.

A well-performing IDS should exhibit high accuracy, indicating its robust overall detection capability, and high True Positive Rate (TPR), demonstrating its efficiency in identifying malicious messages. Conversely, the False Positive Rate (FPR) should remain low, representing a reduced frequency of false alarms. It's important to emphasize that a high TPR and low FPR are pivotal performance metrics for IDSs. This is especially critical in the context of in-vehicle networks [14], where false positives could induce unnecessary alarm and squander resources, including emergency services.

Table 6.2: BNN vs. full-precision NN on CPU

		BNN-1 bit	NN-32 bits
	Inference time /message	0.04 ms	0.12 ms
All Zero ID Attack	FPR	3.75 %	0.08 %
	TPR	97.31 %	99.72 %
	Accuracy	96.53 %	99.82 %
Random ID Attack	FPR	3.32 %	0.12 %
	TPR	98.16 %	99.62 %
	Accuracy	97.84 %	99.92 %
Replay Attack	FPR	8.02 %	0.32 %
	TPR	94.32 %	99.25 %
	Accuracy	93.11 %	99.59 %
Spoofing Attack	FPR	11.03 %	0.88 %
	TPR	88.90 %	99.01 %
	Accuracy	85.12 %	98.89 %
Hybrid Attack	FPR	18.02 %	1.02 %
	TPR	80.23 %	98.02 %
	Accuracy	79.99 %	98.45 %

By following the common practice, for each dataset, we use 70% for training and 30% for testing. Based on our experiments, we notice that the results of each dataset are comparable, thus we report the average of their performance in Table 6.2. It also demonstrates that the applicability of our IDS is not limited to a specific vehicle model. Results in Table 6.2 are calculated over 10 experiments. It can be seen that BNN inference time is about 3 times faster than regular NN. This is an improvement for real-time applications that are time-sensitive. Authors of [49] report that their DNN-based IDS has a processing latency of 2-5 ms per message, which meets the real-time requirements of CAN. Our scheme achieves a shorter processing time.

BNN has a noticeably higher false positive rate compared to the 32-bit NN. For a true positive rate, BNN outperforms the all-zero attack and the random ID attack in comparison to other types of attacks. As for accuracy, BNN can handle the random ID attack better than other attack types in general, but there is still room to improve when compared with full-precision NN in terms of any attack type. The overall performance of BNN on the spoofing attack and the hybrid attack is

not comparable with others, and it might be because those attacks are more complicated than other types of attacks. The proposed BNN-based IDS runs 3 times faster in detection while trading off 2.1%-18.5% accuracy depending on different attack types compared with the 32-bit NN model.

Even though we focus on neural networks for higher accuracy, we also conduct experiments to compare the performance of our scheme with some classical machine learning methods, including decision trees, random forests, and SVM. These classical machine learning methods generally need more human interaction and additional efforts, such as data preprocessing and feature engineering, compared with neural networks. It has been shown that neural networks outperform classical machine learning methods, in both general IDS tasks [121, 134, 23, 50] and CAN bus IDS tasks [49, 102]. We draw the same conclusion through our experiments.

6.3.6.2 Experiment 2: BNN Implementation on FPGA, CPU, and GPU

In this experiment, we compare the performance of BNN-based IDS with different platforms. The same BNN model under the hybrid attack is used on four common hardware platforms: desktop CPU, embedded CPU (eCPU), FPGA, and GPU. We measure the power consumption of the FPGA hardware by connecting a USB power monitor to measure the total power consumption of the board and estimate the power consumption of others based on the hardware specifications, which is accurate enough to use for our purpose [141]. It is noted that the processing delay of the input generator is not included. From the results, not surprisingly, the GPU device is the fastest inference device during our test. Inference on the FPGA device is 2.7 times faster than the CPU and 128 times faster than the eCPU, because of its optimization on the bitwise operation and highly efficient parallel data processing. When we consider the power efficiency (message/sec/watt), FPGAs are the best hardware solution in our experiment, which is 6.3 times than the GPU, 55.8 times than the CPU, and 91.2 times than the eCPU. In practical usage, using powerful GPUs or desktop CPUs is not a feasible solution for ECU setup, therefore, compared with the eCPU platform, which is limited on computation resources, the FPGA platform becomes a recommended solution as a BNN-based IDS inference device.

Table 6.3: BNN - hybrid attack implementation

	FPGA	CPU	eCPU	GPU
Power consumption (watt)	3.1	65	2.2	215
Throughput (message/sec)	55666	20950	434	613002
Efficiency (message/sec/watt)	17957	322	197	2851

6.3.6.3 Experiment 3: BNN Models with Wider/Deeper Structure

To further improve the accuracy performance of the proposed IDS, we explore several BNN model choices with different width and depth settings to understand if and how more complicated network structures can provide better detection accuracy. We evaluate those models from the perspective of effectiveness and efficiency. We also analyze the model size of each model.

Effectiveness Evaluation: Table 6.4 provides the accuracy performance of different model choices. We evaluate these models on the same CPU platform with the same type of attack (hybrid attack) using three metrics - FPR, TPR, and accuracy. The baseline model (3 hidden layers, 1024 neurons in each hidden layer) is the model evaluated in Section 6.3.6.1. Model 1 is a deeper model (5 hidden layers, 1024 neurons in each hidden layer), and Model 2 is a wider model (3 hidden layers, 2048 neurons in each hidden layer). Model 3, it is a deeper & wider BNN model (5 layers, 2048 neurons in each hidden layer). We examine all four models using binary and 32-bit precision options to provide comparisons between binary and full-precision models. We also discuss the model size of these models.

Model 3 provides the most significant accuracy improvement on both binary and 32-bit precision networks compared to other models. However, the cost for Model 3 is also apparent. For binary networks, Model 3 improves 5.67% accuracy performance over the baseline model, while the model size increases from 515 Kb to 2.79 Mb, as seen in Table 6.6.

Increasing width or depth independently can also improve IDS accuracy performance. For

Table 6.4: Effectiveness evaluation - hybrid attack

		baseline model	Model 1	Model 2	Model 3
binary	FPR	18.02%	17.55%	15.20%	12.34%
	TPR	80.23%	81.23%	84.33%	85.42%
	Accuracy	79.99%	81.33%	84.32%	85.66%
32-bit	FPR	1.02%	0.98%	0.94%	0.93%
	TPR	98.02%	98.34%	98.83%	98.88%
	Accuracy	98.45%	98.57%	98.93%	99.12%

binary networks, Model 1 and Model 2 achieve 1.34% and 4.33% accuracy boost respectively, over the baseline model. As a trade-off, these two models' size increases to 787 Kb and 1.76 Mb, respectively.

We also evaluate the effectiveness of each model design choice when the network precision is 32-bit. Since the baseline model has already reached 98.45% accuracy, increasing the width or depth of the model cannot significantly improve accuracy performance. On the other hand, the model size can be expanded multiple times in pace with width and depth adding. Even for the simplest 32-bit model, the baseline model size becomes 15.13 Mb, 29.4 times larger than the binary baseline model size, and 5.4 times larger than the size of the binary Model 3 which is the best performing binary model. This experiment shows that BNN models have an advantage on the model size reduction and demonstrates how we can improve accuracy performance over the baseline model. For different applications, different models can be selected based on the resources they have to meet their needs.

Efficiency Evaluation: We evaluate the efficiency performance of the baseline BNN model, Model 1 (deeper), Model 2 (wider), and Model 3 (deeper & wider) on the same CPU platform under the same type of attack.

As shown in Table 6.5, we measure the inference time/ message for four model design choices. For binary models, the same as the effectiveness evaluation experiment, whether it is 1-bit or 32-bit precision, Model 3 increases the inference time of the baseline model the most. With the model complexity increasing, the inference time required for each message increases. This increasing

Table 6.5: Latency evaluation (inference time/message)

	baseline model	Model 1	Model 2	Model 3
binary	0.04 ms	0.08 ms	0.13 ms	0.22 ms
32-bit	0.12 ms	0.24 ms	0.41 ms	0.67 ms

Table 6.6: Model size of different models

	Binary baseline	Binary Model 1	Binary Model 2	Binary Model 3	32-bit baseline
Size	515 KB	787 KB	1.76 MB	2.79 MB	15.13 MB

inference time needs our attention because real-time embedded systems are usually susceptible to such delays.

The inference time of each binary model is about three times faster than that of the corresponding 32-bit model. This experiment suggests using BNN can effectively decrease processing delay.

6.4 Discussion

As shown in Section 6.3, the proposed IDS can reduce the detection latency while maintaining acceptable detection rates, and the accuracy can be further improved. It is reasonable to expect that the proposed IDS can be available for real-time detection. However, there is a trade-off to consider: the balance between detection performance and processing delay. Our findings suggest that while our BNN-based IDS provides rapid detection, the detection rates and accuracy are lower than that of full-precision models across all types of attacks. Meanwhile, false positive rates remain higher than those observed in full-precision counterparts.

To enhance performance, a few strategies can be considered. One potential approach is the adoption of more complex neural network architectures that may better model the complexity and intricacies of attack patterns. Another strategy can involve the incorporation of features derived from the semantic interpretation of CAN traffic data. This would enable a deeper understanding of the underlying data, which can help improve the robustness of the IDS. However, it is important to

note an inherent limitation affecting all supervised learning models, including neural networks used in IDSs, such as RNNs, and CNNs [102, 50, 49]. These models may struggle to identify unlearned types of attacks, as their capacity to detect new threats is inherently bound by the data they've been trained on. This challenge underscores the need for further research to better equip these systems to detect previously unseen or novel attacks, further enhancing the utility and reliability of IDSs.

On the bright side, given that it strikes between processing speed and detection performance, our BNN-based IDS exhibits potential for real-time intrusion detection tasks. Its fast detection and the ability to maintain satisfactory detection performance make it a promising candidate for real-time applications where immediate threat identification and response are crucial. This introduces a new frontier for the practical applicability of IDSs in real-world scenarios, expanding their utility in automotive cybersecurity.

6.5 Conclusion

In this chapter, we propose a BNN-based IDS to secure the in-vehicle CAN bus and protect vehicles from malicious intrusions. The proposed IDS utilizes BNN to accelerate intrusion detection while reducing detection latency, memory request, and system power consumption. The IDS, which is designed to suit CAN data, can process the raw CAN data without additional prepossessing or hand-designed features. It learns from sequential patterns from CAN traffic rather than individual CAN message. We evaluate the proposed IDS on datasets collected from four real vehicles of different models and manufacturers, which also helps validate that our IDS's applicability is not limited to a specific vehicle model. Experimental results demonstrate that the IDS is able to detect malicious behaviors with lower latency (3 times faster) compared to full-precision neural network-based IDS with sacrificing the accuracy (2.1%-18.5% depending on the types of attacks) on the same CPU platform. And the model runs 128 times faster on FPGAs than an embedded CPU. Moreover, the experiments show a BNN model with a wider or deeper structure can achieve better accuracy, but as a trade-off, it increases detection latency and model size to varying degrees. In

addition, experimental results present that the BNN model provides a significant advantage in the model size reduction compared to 32-bit counterparts. Therefore, different applications can select the proper model according to their needs and resources.

Our study can serve as a proof-of-concept that our BNN-based IDS has the potential to enhance in-vehicle security significantly, especially considering the embedded environments of in-vehicle networks.

CHAPTER 7

Efficient and Effective In-Vehicle Intrusion Detection System using Binarized Convolutional Neural Network

7.1 Introduction

As presented in Chapter 6, given the demand for real-time communications of in-vehicle networks, an IDS for in-vehicle networks should be fast in detection time and small in memory size [138, 139]. That chapter introduces a BNN-based IDS, aimed at enhancing the computational efficiency of our machine learning-based IDS. However, the overall accuracy of BNN-based IDS needs to be improved to better meet the demands of securing in-vehicle networks. Firstly, that design does not take full advantage of the temporal and spatial features of messages that can contribute to a higher accuracy [98]. Secondly, that design uses a binary neural network based on a shallow multi-layer perceptron (MLP) architecture, which may not generalize well enough to handle complex CAN traffic.

This chapter proposes an IDS that consists of two main components: an input generator and a Binarized Convolutional Neural Network (BCNN) model. The input generator aims to enable the proposed IDS to utilize temporal and spatial features of CAN traffic for higher detection rates. We design the input generator based on a processing framework called DeepInsight [98] to convert CAN messages from a feature vector into a well-organized image form, which can provide temporal

and spatial features of CAN messages to the BCNN model. The input generator transforms the CAN traffic data according to their similarity. Moreover, it has the potential to reduce the processing latency of the proposed IDS, because it can flexibly set an appropriate input image size, thereby reducing the input dimensionality.

For the second component, we propose to utilize BCNN to process output images from the first component. Among various machine learning methods used to build CAN IDS, CNN is powerful and easy to use [102]. CNNs can deliver notable performance by extracting features from raw inputs through convolutions and pooling, eliminating the need for supplementary feature engineering techniques. However, like other neural networks, it can be challenging to use CNN for CAN IDS due to the high latency and considerable memory requests. In order to maintain the advantages of CNN, we utilize BCNN with binarized activations and weights, which can accelerate intrusion detection and reduce memory requests compared with the corresponding full-precision models. Especially, BCNN has a similar structure to CNN, which allows BCNN also has the power to learn and process high-order statistics and nonlinear correlations of inputs to provide better performance. Note that applying binarization to full-precision CNN may significantly degrade IDS detection accuracy without an appropriate design, due to the trade-off between detection performance and processing latency. In this work, we design an IDS with a BCNN model that can provide better accuracy than the IDS with a BNN model presented in the previous chapter, and we explore QuickNet [3] architecture with its different configurations for our proposed IDS to further improve the accuracy of the original BCNN model. It is noticed that QuickNet-Large can provide the best accuracy with less model size required than the original BCNN model.

We evaluate the proposed model using datasets from four vehicles across different manufacturers. Real CAN data collected from actual vehicles is essential for evaluating CAN IDS performance because simulated CAN datasets are unrealistic and impractical, and real-world factors such as temperature changes can affect CAN messages [14]. Also, as the actual CAN ID and data depend on the manufacturer, it is crucial to demonstrate the applicability of our scheme on different car models. Experimental results show that the proposed IDS achieves a satisfactory detection rate and

low latency.

Contributions. This study makes the following contributions:

- To the best of our knowledge, this work is the first to utilize BCNN for IDS in in-vehicle networks. Additionally, we develop an input generator that converts CAN traffic into temporal sequential patterns and spatially local correlations of CAN messages for the proposed BCNN model.
- The proposed BCNN-based IDS achieves four times faster processing compared to a full-precision CNN-based IDS while maintaining satisfactory accuracy. It also requires less memory, which is crucial for resource-constrained embedded devices in in-vehicle network environments.
- We investigate a variety of binarized CNN architectures, ranging from AlexNet to QuickNet with different configurations for our proposed IDS design. This showcases the flexibility of our IDS and its potential for higher detection rates, which can meet varying resource constraints and accuracy requirements.
- We conduct extensive evaluations using real CAN traces from four real vehicles, in contrast to most prior studies that rely on simulated data or data from no more than two vehicles.

Organization. This chapter is organized as follows. Section 7.2 provides the design of the proposed IDS. Section 7.3 details the insights into the implementation and evaluation metrics of the proposed scheme. Section 7.4 discusses the proposed IDS. Section 7.5 concludes this chapter.

7.2 Our Design

We aim to build an in-vehicle network IDS to protect CAN bus from attacks by detecting malicious messages in the bus with low latency as well as high accuracy. It is supposed to be deployed on a central gateway or installed on an ECU as a node on a protected CAN bus to monitor traffic.

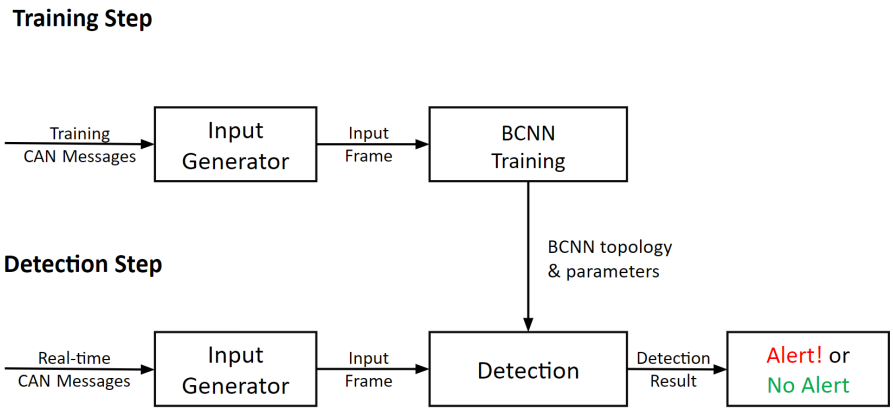

Figure 7.1: Workflow of the proposed BCNN-based IDS

As shown in Figure 7.1, the proposed BCNN-based IDS scheme consists of two main steps: 1) the training step and 2) the detection step. In the training step, to enable BCNN to learn temporal and spatial patterns of CAN messages, we design an input generator that converts consecutive CAN messages to image inputs and attaches corresponding labels to construct input frames. If one or more attack messages are in an input frame, its label is set to malicious. Training a classifier is considered time-consuming, and it can be done offline. In the detection step, the input generator transforms the current traffic message as input without labels. These inputs are processed by the trained BCNN model obtained in the previous step. This model can be deployed in ECUs, including some general-purpose hardware such as CPUs, GPUs, or FPGAs [138]. The output is a prediction of whether an input is malicious or not. An alarm will be raised if malicious input is detected.

7.2.1 Input Generator for BCNN-based IDS

To save additional processing time, our objective is to process raw CAN traffic data without any hand-designed features that can introduce extra latency. We propose an input generator to allow the BCNN model to learn CAN traffic. The generator transforms consecutive CAN messages into well-organized images, enabling the model to learn and further identify malicious messages in the traffic.

7.2.1.1 Limitations of Prior Work

Researchers have proposed different methods to build image inputs for CNN-based IDS to protect CAN. Traffic data patterns are changed under attacks [124] and can be exploited as detection features. For example, every 29 consecutive CAN message IDs are assembled directly into image inputs and enable a CNN model to learn for detection [102]. However, this design does not cover the data section of CAN messages which cannot detect CAN messages with valid IDs and tampered data. [138] proposes a design in which both ID and data sections of CAN messages are included. Although such a design covers the data section, this scheme does not provide sufficient accuracy. Temporal or spatial location dependencies [19, 98] that may contribute to high accuracy are not taken into account in either work.

Recurrence plots are used to generate images from the CAN messages to capture the temporal dependency [19]. A recurrence plot is a graph of a square matrix where each element corresponds to times at which a state of a dynamical system recurs. The majority of CAN messages are sent to the bus periodically [14]. Thus, the visual appearance of a CAN messages recurrence plot can offer temporal dependency on the dynamics of the system. In [19], each recurrence plot is generated by 128 consecutive CAN IDs without data sections. Figure 7.2, (a) and (b) show example recurrence plots using one of our datasets under the spoofing attack. The normal state and the attack state are difficult to distinguish, which means that there is not enough information to train the model, which also makes intrusion detection challenging. We investigate a new similar method that generates recurrence plots based on both ID and data sections. Examples generated by the same CAN messages and attacks are shown in Figure 7.2, (c), and (d). The problem still exists. We do not use recurrence plots for two main reasons: 1) they cannot provide enough meaningful information for training the model, making IDS ineffective against some common attack types; 2) they cannot consider that some vehicles have aperiodic CAN messages [14], which will bring a large number of false positives.

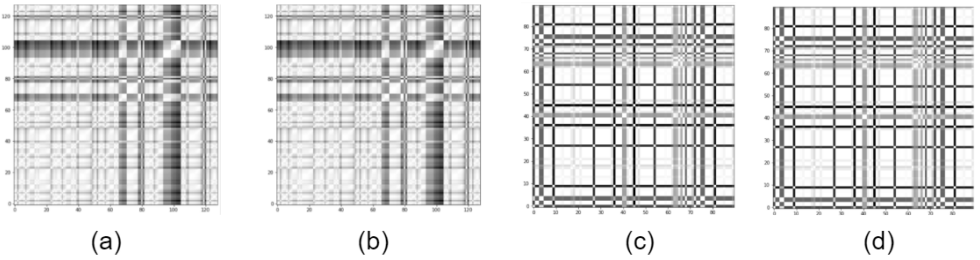

Figure 7.2: Recurrence plots of CAN traffic using 1) only ID, without attack (a) and with attack (b); 2) ID and data, without attack (c) and with attack (d)

7.2.1.2 Our Input Generator

To achieve a high accuracy performance, we leverage temporal sequential patterns and spatially local correlations of CAN messages. We design an input generator that is based on DeepInsight [98], which transforms consecutive CAN messages into well-organized images, considering potential spatially local correlations of data. As consecutive messages also represent temporal sequential patterns, both temporal and spatial features are involved, and thus the input generator enables the deep learning model effectively to take advantage of both features of consecutive CAN messages.

Each image input is constructed by placing similar elements together and dissimilar ones apart, which enables the collective use of neighboring elements. For a standard CAN data format, it contains an 11-bit ID and 8-byte data, i.e., a message can be represented in 10 bytes in total: $d^0, d^1, d^2, ..., d^8, d^9$. Therefore, for m consecutive CAN messages, they construct a feature vector $V = \{d_0^0, d_0^1, d_0^2, ..., d_m^8, d_m^9\}$, which contains the number of elements m times 10. The number of consecutive CAN messages should be chosen carefully. Too small m may not provide enough information, including the temporal feature, while too large m may significantly increase processing time. To find the appropriate m, we conduct experiments using datasets collected from four vehicles and set it to 100. Note that the appropriate value of m may vary for CAN data from other vehicle models.

Before image transformation, values should be normalized because each layer of an image containing 256 shades is normalized in the range of [0,1]. Based on the experiments we performed

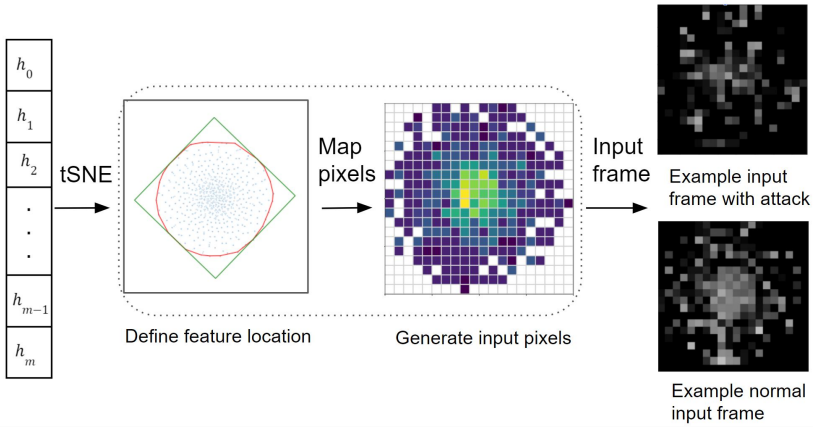

Figure 7.3: Workflow for converting CAN messages into image pixels

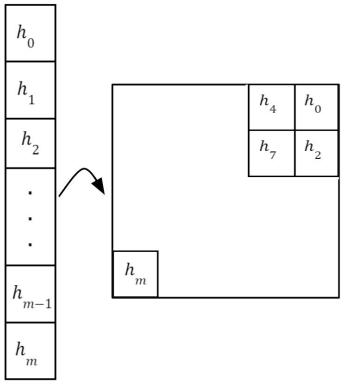

Figure 7.4: Example of converting feature vector to feature matrix

and the fact that each feature can be considered independent in CAN data, we normalize each feature by its minimum and maximum values. Figure 7.3 illustrates a workflow of our input generator. There are two main steps: 1) define feature location and 2) generate input pixels.

In the first step, the training data is used to find each feature position based on the similarity between the features. If the training dataset T contains n instances, $T = \{V_0, V_1, ..., V_{n-1}\}$, where V_i represents a feature vector including m consecutive CAN messages. After that, T is transposed to a feature set $D = \{h_0, h_1, ..., h_{m-1}\}$, where $h_i = \{d_0^i, d_1^i, ..., d_{n-1}^i\}$ represents a feature including n training instances. Then, we apply t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm [117, 98] on this feature set D to define the location of features based on similarity. t-SNE is a

dimensionality reduction technique that is chosen because it provides better results than other options based on our analysis of the four CAN datasets we collected. Other similarity measuring techniques or dimensionality reduction techniques, including kernel principal component analysis (kPCA) may be considered for other vehicle models [98]. As illustrated in Figure 7.4, T is transformed into a feature matrix D, which can present spatial correlations compared with feature vectors. The similarity of features decides the location of each feature in the Cartesian coordinates. For example, h_0, h_2, h_4, h_7 are closer to each other. Then, we find the smallest region containing all points by applying the convex hull algorithm [15], which reduces redundant regions in the input that can cause additional unnecessary processing. Next, the frame is rotated to a horizontal position, as CNN generally prefers an image in a horizontal over a vertical form. The output of the convex hull algorithm and the rotation result are shown in red and green, respectively.

In the second step, input pixels are generated. Cartesian coordinates are converted to pixels by averaging certain features according to the image size. A feature is averaged and placed at the same location when multiple features have the same location in the pixel frame. If the image size is very small, many features can overlap, resulting in low-quality images leading to low accuracy; if the image size is very large, the image representation is too large, which could cause significant unnecessary processing delays. We set the resolution to 20 x 20 to reduce the number of computation parameters in CNN models based on the reporting result from [127]. Finally, current CAN traffic can be mapped to pixel locations and produce input frames. Example input frames with one of our datasets under normal state and spoofing attack are also presented in Figure 7.3.

Our input generator enables our BCNN model to learn the temporal and spatial characteristics of CAN traffic for high accuracy. Unlike many existing CAN IDSs [102, 14] that only focus on the ID field of CAN messages for intrusion detection, our IDS can cover complete information of each message. Otherwise, they cannot effectively detect malicious messages, such as spoofing messages, constructed with valid IDs and malicious data fields. In addition, this input generator can also help reduce the dimension of input for the subsequent machine learning model when applying a sufficiently small resolution that results in acceptable performance, which can further reduce the

processing delay in detection.

7.2.2 Binarized Convolutional Neural Network

In the continuously evolving landscape of automotive cybersecurity, intrusion detection has become paramount for maintaining secure communication networks. Applying Machine Learning (ML) algorithms, especially Neural Networks (NNs), has shown remarkable promise in this context. While BNNs offer some advantages, such as decreased computation time and lower power consumption, they are frequently hampered by lower accuracy levels than their full-precision counterparts. This limitation is particularly noticeable when dealing with complex intrusion patterns.

As one of the deep learning-based pattern recognition methods, CNN is able to efficiently and effectively learn features from input CAN messages than classical neural networks [129, 102]. In addition, as another advantage of CNNs, the same convolutional kernels are shared, which would reduce the number of parameters and calculation amount of training once greatly. Thus it can more quickly identify attack type of traffic data. However, CNNs generally involve high computational costs and large memory accesses resulting in large power consumption [8, 17]. In view of the limited ECU memory and the requirement for real-time communication, the CAN IDS requires fast detection and small model size. We propose using BCNNs for intrusion detection to protect CAN from cyberattacks. Unlike common CNNs, with binarization, BCNNs use binarized weights and activations instead of full precisions. Computation costs can be reduced significantly with improved efficiency by using bit-wise operations.

A BCNN model has a similar structure to a full-precision CNN model, comprising an input layer, a fully-connected layer, and an output layer, along with varying numbers of convolutional and pooling layers [29]. The key component is the convolutional layer containing a set of kernels. In the binarized network, during the forward propagation stage, floating-point weights w and activations a are replaced by w_b and a_b that are the tensor of binary weights and binary activations, with the corresponding scalars α and β. The definition of the binarization functions $Q_w(w)$ and $Q_a(a)$ are given as follows:

$$Q_w(w) = \alpha w_b, Q_a(a) = \beta a_b \tag{7.1}$$

In full-precision convolutional neural networks, the basic operation can be expressed as [93]:

$$z = \sigma(w \otimes a) \tag{7.2}$$

, where z is the output tensor and $\otimes$ represents the convolution operation.

A BCNN can be reformulated in the regular CNN format as follows, with the binarized weights and activations:

$$z = \sigma(Q_w(w) \otimes Q_a(a)) = \sigma(\alpha\beta(w_b \odot a_b)) \tag{7.3}$$

, where $\odot$ is an inner product for vectors that operate using XNOR-Bitcount [93].

The binarization method converting 32-bit to binary values uses the same way in BNN, which can refer to Section 6.2.2.1.

BCNN Model Configuration: We design a BCNN model that takes input frames from the input generator and outputs a binary decision indicating whether or not there are any malicious messages among the messages. Taking a simple and straightforward CNN architecture AlexNet, we set up the proposed BCNN model built with five binarized convolutional layers and three fully connected layers [38]. Relu is applied after each convolutional and fully connected layer except the output layer. To avoid overfitting issues, we use batch normalization layers that standardize the inputs to a layer for each mini-batch, and use dropout layers with a 15% dropout rate. Both techniques help reduce overfitting and improve the generalization of the model. We adopt training strategies described in [17], including shift-based batch normalizing and AdaMax [53]. In the detection step, because almost multiple-accumulate operations in the network are converted into binary operations, the BCNN model can process CAN frames at a fast speed while utilizing a small number of memory resources.

7.2.3 Accuracy Improved BCNN Model – QuickNet

The proposed baseline BCNN improves latency and memory utilization from full-precision CNNs; however, as a trade-off, the IDS accuracy performance is degraded because of parameter binarization. In practice, accuracy performance is critical when applying an IDS for in-vehicle networks. Given that fact, we need to find a way to improve the accuracy of baseline BCNN while still maintaining the advantage of speed and size.

We explore some other design options based on the baseline BCNN. Bethge et al. [5] and Martinez et al. [70] discuss bottlenecks in the binarized network structure, and they identify two factors that largely impact the accuracy performance of BCNN: the first full-precision layer and other non-binary operations. Bannink et al. [3] propose a network called QuickNet which addresses the two performance bottleneck issues and successfully benchmarks the popular CNN-based binarized model. From their experimental results, QuickNet can outperform other BCNNs in terms of accuracy and latency in image classification tasks using ImageNet dataset.

Thanks to the input generator that makes it possible to process CAN messages in any CNN model. To our best knowledge, we are the first team to utilize BCNN to IDS for in-vehicle networks, and furthermore, we adopt the STOA BCNN design – QuickNet in our design to make it work well with the proposed IDS.

Addressing the performance bottleneck, QuickNet improves the efficiency of the full-precision first layer and meanwhile achieves competitive accuracy. In the first layer, there is a small convolution of size 3×3 with 16 filters, then follow by a depthwise separable convolution to increase the feature size. As a result, the spatial resolution from the input generator of size 20x20 decreases to 10x10 using striding. For the transition block, as another performance bottleneck, we apply a transition block in hidden layers, which is made with a 3×3 antialiased max pooling. The max pooling layer, a strided depthwise convolution layer with a fixed blurring kernel, and a 1×1 full precision convolution with filters work together to increase the feature size.

To make QuickNet work with the proposed CAN IDS, the following steps are taken in this work. We firstly modify the fully connected layers, which are used to work for 1000 categories

image classification task, and now output 0 or 1 for "normal message" or "malicious message". Secondly, according to previous CAN IDS research [136], compared to regular images, image frames generated by CAN messages do not need too large or complex deep learning architectures to learn their features. We build a QuickNet-based model with only half the transition block than the original QuickNet, and we find that more redundant layers cannot contribute to accuracy but cause issues like overfitting. Lastly, we follow a similar strategy from [3] to train the model with some tune parameters which work better for the proposed IDS. 1000 epochs are trained from scratch on a GPU, with a batch size of 512. Adam optimizer is used with an initial learning rate of 0.005. The Binarized weights and activations are processed with the straight-through estimator [17]. SGD (stochastic gradient descent) is applied for full-precision variables with a momentum of 0.9 and a learning rate of 0.1. We also explore variants of QuickNet with different sizes of widths and depths. A half-wide design called QuickNet-Small and a double-deep design called QuickLarge are designed and evaluated. The experimental section discusses in detail how the width and depth sizes affect inference time and accuracy performance.

7.3 Implementation and Evaluation

7.3.1 Experimental Datasets

We collect CAN data from the on-board diagnostics II (OBD-II) interface, a standard diagnostic port, by employing the CANalyst-II device as introduced in Section 4.2. The OBD-II port provides diagnostics, emission measurements, and other information, such as engine control, body control, and chassis control information. As we used the representative datasets in Chapter 6, we use the same datasets to evaluate the proposed IDS. This allows for a comparison of the experimental outcomes with those obtained by other approaches and helps ensure consistency. Details on each dataset used in the evaluation are provided in Table 7.1. It is noted that the proposed IDS is tested on four datasets from different vehicle models and manufacturers to demonstrate the proposed IDS's applicability across various vehicle models.

Table 7.1: Experimental datasets

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Vehicle Model	Honda Accord 2006	Honda Civic 2018	Ford Fusion 2013	Chevrolet Volt 2013
Number of Messages	243,762	1,806,780	2,158,201	4,536,342
Number of Unique IDs	13	54	79	106

7.3.2 Attack Strategy

An attacker launches attacks by injecting malicious CAN messages into the bus. These messages can change the normal CAN traffic density. To ensure that malicious messages can be received, the attacker typically sends malicious messages more frequently than normal messages. In a common attack scenario, the number of malicious messages should be at least double that of normal messages, though, in practice, this ratio may rise significantly, potentially reaching 20 to 100 times higher [73].

Considering our adversary model, real-world attacks, and existing research on attack strategies [73, 14, 102], we perform 100, 200, 200, 300 intrusions on Dataset 1, 2, 3, and 4, respectively, depending on their size. Each intrusion can last three to four seconds, and during each intrusion, a certain type of attack occurs. For the *all zero ID* attack, we inject a CAN message with 11-zero-bit ID every 0.3 ms. For the *random ID* attack, we follow a similar way to the all zero ID attack with randomized ID and data messages. For both *replay* and *spoofing* attacks, we inject malicious messages every 0.8 ms, with replay attacks using valid, previously transmitted messages and spoofing attacks utilizing messages containing valid IDs with altered data. For hybrid attacks, we randomly choose to inject a malicious message from all four types of attacks every 0.3 ms.

7.3.3 Experimental Setup

We design and conduct experiments on CPU and GPU to evaluate the effectiveness and efficiency of our BCNN-based IDS. Python version 3 and Larq Compute Engine [3] built on top of TensorFlow are used for implementation. Models are trained on a Linux machine running Ubuntu 20.04.2 LTS with a CPU (AMD Ryzen 3700x with 8 cores and 16 threads, base clock at 3.6GHz) and GPU (Nvidia RTX 2070 Super, 8GB DDR6 RAM). We use the same device to demonstrate the performance of our IDS in terms of accuracy and inference time in the experiment.

7.3.4 Experimental Results

7.3.4.1 Experiment 1: BCNN-based IDS under Different Attacks

To evaluate the effectiveness and efficiency of our IDS, we use the same metrics as in Chapter 6: accuracy, true positive rate (TPR, or detection rate), and false positive rate (FPR).

These performance metrics are essential for evaluating IDSs. Accuracy represents the overall correct detection rate, TPR indicates the proportion of correctly identified malicious messages, and FPR corresponds to the fraction of non-malicious messages mistakenly flagged as malicious. For an IDS to perform efficiently, it should exhibit high accuracy and TPR but maintain a low FPR. Given the critical context of in-vehicle networks [14], it is particularly important to keep FPR low to prevent unnecessary panic and wastage of essential resources, such as emergency services. Hence, these metrics are often the principal criteria for assessing IDS performance.

The performance of BCNN-based IDS is evaluated under different types of attacks using four datasets, as shown in Table 7.1. For each dataset, allocating 70% for training and 30% for testing with results based on 10 repeated experiments. Based on experimental results, we observe that the results across each dataset are comparable, so we report the mean in Table 7.2. These results also demonstrate that the applicability of the proposed IDS has a good potential to be applied to different vehicles.

The accuracy rates under the random ID attack, the all zero ID attack, the replay attack, and the

Table 7.2: BCNN-based IDS under different attacks

	Accuracy	FPR	TPR
Random ID Attack	98.46%	2.86%	98.95%
All Zero ID Attack	98.01%	3.68%	98.64%
Replay Attack	96.82%	5.03%	97.51%
Spoofing Attack	94.19%	10.18%	95.74%
Hybrid Attack	89.51%	17.07%	92.16%

spoofing attack are over 94.19%, while the performance under the hybrid attack is not comparable with others. This may be because those attacks are more complicated than other attack types. In the following experiment, we focus on detecting the hybrid attack, which we consider to be the most complicated one.

7.3.4.2 Experiment 2: Effectiveness Evaluation on Input Generator

In this experiment, we assess the effectiveness of the proposed input generator, a novel data preprocessing module for any CNN-based IDS. The results, as outlined in Table 7.3, are all obtained under uniform conditions, using the same hardware platform and the same attack model. Furthermore, we employ an identical BCNN model, which is used after input generation.

Our study is conducted in two stages. Firstly, we examine the IDS performance without the aid of an input generator. Referring to the design in [139], we aggregate 128 consecutive CAN messages and reconfigure them into the grid format. This method, henceforth denotes as "without input generator" (w/o IG), yields an IDS accuracy of 86.64% with a negligible processing delay.

Subsequently, we incorporate our proposed input generator into the system. This module designates as "with input generator" (w/ IG), and also processes the same number of CAN messages as the former method. Upon implementation of the IG, the IDS accuracy significantly increases to 89.51%, with a slight processing delay of 0.1ms.

Despite the minor increase in processing delay when the IG is used, the benefits in terms of improved IDS accuracy are significant. This suggests that the slight delay introduced by the input

generator is acceptable, given the marked improvement in detection accuracy. Thus, integrating the proposed input generator into the IDS serves as a valuable addition to the system, underscoring the potential of our data preprocessing module in enhancing the overall effectiveness of any CNN-based IDS.

Table 7.3: Input generator effectiveness

	Accuracy	Processing Delay
w/o IG	86.64%	0 ms
w/ IG	89.51%	0.1 ms

7.3.4.3 Experiment 3: Effectiveness/Efficiency Evaluation

We benchmark the proposed BCNN-based IDS against two other deep learning-based IDS designs: 32-bit CNN and BNN. These three neural network models represent different approaches within the proposed scheme. 32-bit CNN, based on AlexNet[55], is a powerful full-precision CNN. BNN[139], utilizing a multiple-layer perceptron architecture, employs the same binarization approach as BCNN to minimize model size and latency, but lacks convolutional processes.

They share the same overall scheme design shown in Figure 7.1: a set of raw CAN data gets processed in the input generator, then framed CAN images are processed in neural networks. CNN-based IDS and BCNN-based IDS use the input generator proposed in this work, which is specialized for CNN-based architecture. Three different models are evaluated under the same hybrid attack. The output of neural networks is a true or false prediction, and we report accuracy in Table 7.4. The size of the neural network model and inference time per frame (100 consecutive CAN messages per frame) are also presented for efficiency evaluation. It can be seen that the BCNN model has a significantly smaller size. Furthermore, F1 score is introduced to measure how well different classification models can correctly identify positive and negative cases, and a higher score indicates better performance. F1 score is calculated as defined in Equation 4.9.

The results demonstrate that the 32-bit CNN achieves the highest accuracy among the compared

models, but its model size is 81.9 times larger than the BNN and 30.5 times larger than the BCNN. Meanwhile, it has a longer inference time compared to the other models. In contrast, the BNN, with its simpler architecture, has a smaller size and faster inference time but only reaches 79.9% detection accuracy under hybrid attacks. This performance is 80.6% of the full-precision CNN-based IDS. The BCNN outperforms the BNN in accuracy while significantly reducing both model size and inference time compared to the full-precision CNN, making it a more efficient and effective solution for intrusion detection.

Table 7.4: Effectiveness/Efficiency of BCNN scheme

Model Design	Model Size	Accuracy	F1 score	Inference time (per frame)
32-bit CNN	228.45 MB	99.29%	99.50%	2.4 ms
BNN [139]	2.79 MB	79.99%	85.82%	0.4 ms
BCNN	7.49 MB	89.51%	92.61%	0.6 ms

7.3.4.4 Experiment 4: Improved BCNN Models

In this experiment, we discuss the accuracy improvement strategies of the proposed IDS that is based on BCNN, a binary-precision AlexNet model. As presented in Table 7.5, it is called BCNN baseline. There are three options that are based on the same improved CNN architecture, called QuickNet [3]. We explore the variant with different width and depth settings to understand if and how more complicated network structures can provide better detection accuracy. We evaluated these models considering both their effectiveness in detecting intrusions and their computational efficiency. The QuickNet-Small is based on QuickNet, and has only half of the width in channels. On the contrary, the QuickNet-Large is also derived from QuickNet, but it composes of 1.5 times more hidden layers by repeating the redundant blocks in hidden layers.

Results show that any QuickNet model outperforms the baseline in terms of model size, accuracy, and inference time. As illustrated in Table 7.4 and Table 7.5, QuickNet-Large based-IDS detection accuracy is 91.88%, which can achieve 92.54% accuracy performance of the 32-bit CNN based-

Table 7.5: Further improved BCNN design

	Model Size	Accuracy	Inference time (per frame)
BCNN baseline	7.49 MB	89.51%	0.6 ms
QuickNet Small	4.01 MB	90.32%	0.5 ms
QuickNet	4.36 MB	91.41%	0.5 ms
QuickNet Large	5.58 MB	91.88%	0.5 ms

IDS. Meanwhile, QuickNet-Large based-IDS is 40.9 times smaller in model size and 4.8 times faster in processing time. Table 7.5 also indicates that the performance of BCNN-based IDS can be further improved by only replacing neural networks in the proposed IDS scheme. In the future, when there is a need to adjust, fine-tune, or improve the overall performance of the proposed IDS scheme, the input generator, and the binarized convolutional neural networks can be modified independently in this framework to meet different requirements.

7.4 Discussion

In the quest for efficient and accurate IDS, our study explores several variants of BCNN. Our efforts to improve upon the basic AlexNet model have shown promising results, confirming that the proposed BCNN-based IDS can further enhance its efficiency and accuracy. The experiments encompassed various modifications and fine-tunings of the BCNN architecture, underlining its versatility and the potential for performance improvements through innovative adaptations.

However, while the BCNN-based IDS has demonstrated its capacity to detect various attack types, including hybrid ones, its performance may not match the accuracy and efficiency of rule-based IDS systems under predefined rules. For instance, in the scenario of an ”all zero attack,” a rule-based IDS can offer superior detection rates. This is because the rule-based IDS is explicitly designed to recognize such predefined patterns, whereas machine learning models learn from a more general distribution of the data, which might not always prioritize such specific attack patterns.

Nonetheless, our two-stage IDS framework can effectively bridge this gap. In the first stage, we

deploy a rule-based IDS to swiftly and accurately detect attacks under predefined rules. The second stage engages the BCNN model, which can handle a broader spectrum of attack types, including complex or hybrid ones.

This two-stage setup enables us to leverage the strengths of both approaches. The rule-based IDS excels in identifying known attack patterns with high precision, while the BCNN model adds robustness against novel or complex attacks that are beyond the scope of predefined rules. The flexible nature of the two-stage IDS also allows for other machine learning models to be used in conjunction with the rule-based IDS, adapting to the specific needs of the system or network it protects.

In conclusion, our research suggests that a hybrid approach, combining rule-based and machine learning techniques, offers a balanced and flexible solution to intrusion detection. By exploiting the strengths of both methodologies, we can build a more comprehensive and adaptable IDS, capable of dealing with a broad range of attack scenarios while maintaining high detection performance.

7.5 Conclusion

This chapter has demonstrated, for the first time, that an extremely lightweight CNN model, BCNN can be used for CAN IDS. This study will serve as a base for future research into the potential of BCNN-based IDS schemes to significantly enhance in-vehicle security, especially considering the embedded environment of in-vehicle networks. The proposesd IDS composes of an input generator and a Binarized CNN model, and this IDS is designed to suit CAN data. It can process the raw CAN data without additional prepossessing and hand-designed features. The input generator helps learn from the temporal and sequential patterns, and spatially local correlation of CAN traffic instead of an individual CAN message. The BCNN model is able to process features from CAN frames and detect attack messages. The proposed BCNN-based IDS is faster and more compact than common 32-bit deep learning-based IDSs, and it is capable of detecting various attacks with satisfying accuracy. Experimental results demonstrate that even in the most complicated attacking scenario -

the hybrid attack, the QuickNet-Large based IDS which is 40.9 times smaller in model size and 4.8 times faster in processing time can achieve 92.54% accuracy performance of a full-precision CNN IDS. Furthermore, we also present different BCNN models that can be selected according to the specific needs of certain applications.

CHAPTER 8

Conclusion and Future Work

The increasing integration of software and electronic components in modern vehicles, along with their external communications, has introduced new security vulnerabilities and risks. In-vehicle networks, responsible for interconnecting various components such as sensors and electrical units, have become prime targets for attackers with various purposes, including disrupting normal operations or even gaining control over the attacked vehicle. Controller Area Network (CAN) inherently lacks sufficient security features, making it vulnerable to various attacks. Given that the security of vehicles directly impacts the safety of drivers and passengers, securing in-vehicle networks is critical to ensuring the overall safety of the vehicle.

By analyzing in-vehicle networks and existing Intrusion Detection System (IDS) schemes, it has been noticed that numerous schemes lack the essential features required to secure in-vehicle networks. This dissertation proposes novel IDS schemes to secure in-vehicle networks effectively and efficiently. In this chapter, we summarize the contributions and outline future work.

8.1 Conclusion

This dissertation contributes to the field of in-vehicle network security, particularly in the design of IDS. This research explores IDS design for efficient and effective intrusion detection in in-vehicle networks operating in resource-constrained environments with real-time demands. By addressing the challenges outlined in Section 1.2, this dissertation makes the following main contributions:

- **An Empirical Comparative Study on IDS for CAN**

  Many IDS schemes have been developed to protect in-vehicle networks, but evaluating and comparing their effectiveness can be challenging due to varying adversary models, datasets, and evaluation metrics. To gain a comprehensive understanding of their performance, it is critical to evaluate these systems using the same criteria. In Chapter 4, we conduct a comparative study of various IDSs for CAN, ensuring a fair comparison by using the same adversary models, datasets, and evaluation metrics across all systems. The results of this study reveal the strengths and weaknesses of each scheme, enhancing our understanding of CAN IDSs. Insights from this research can guide future work and facilitate the development of robust and efficient IDS, making a significant contribution to the field of automotive cybersecurity.

- **A Two-stage IDS Framework for Efficient and Accurate Intrusion Detection**

  Existing IDS schemes are either rule-based or machine learning-based. Rule-based approaches offer quick detection but may have low detection rates under certain types of attacks. On the other hand, machine learning-based IDS can achieve higher detection rates but come with computational expenses and processing delays. Additionally, both approaches have limitations in detecting certain types of attacks. In Chapter 5, we propose a hybrid approach for efficient and accurate intrusion detection for in-vehicle networks. The hybrid IDS addresses the limitations of existing approaches by combining the benefits of rule-based and machine learning-based IDSs. By leveraging machine learning methods, the system achieves high detection rates, while the computational requirements are kept low by offsetting the detection with a rule-based component. Furthermore, the proposed IDS is a scalable framework that supports future updates and allows individual components to be replaced with alternative rules or machine learning algorithms as needed. The effectiveness and efficiency of the proposed hybrid IDS are evaluated through experiments conducted on CAN data collected from four different real vehicles, demonstrating the capability to detect malicious behaviors

across various vehicles. This study serves as a proof-of-concept, showcasing the potential of the hybrid IDS model to enhance in-vehicle security significantly.

- **Accelerating Intrusion Detection Using Binarized Neural Network**

  IDSs based on advanced deep learning techniques, such as Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), have been proposed to protect the CAN bus. However, these models often introduce significant latency, require substantial memory resources, and result in high energy consumption. To address these challenges, in Chapter 6, we explore the utilization of BNN and hardware-based acceleration to accelerate intrusion detection while reducing memory demands. BNN employs binary values for activations and weights, offering faster computation, reduced memory cost, and lower energy consumption compared to full-precision models. Furthermore, leveraging FPGAs allows for additional acceleration of BNN by minimizing hardware consumption. The proposed scheme, along with FPGA-based acceleration, offers a promising solution for enhancing intrusion detection in in-vehicle networks. It effectively addresses latency, memory, and energy consumption issues associated with deep learning-based IDSs, providing accelerated detection capabilities while maintaining acceptable detection rates. The results of this study highlight the potential of using BNN and FPGA techniques to facilitate efficient intrusion detection and enhance the security of in-vehicle networks.

- **Efficient and Effective In-Vehicle Intrusion Detection System Using Binarized Convolutional Neural Network**

  To protect CAN from potential attacks, IDSs have been proposed. However, IDSs that utilize advanced deep learning techniques often suffer from performance drawbacks, including long processing times and large memory requirements. Existing BNN-based IDSs, suggested as a solution, have exhibited low accuracy in securing CAN. To overcome these limitations, in Chapter 7, a novel IDS utilizing BCNN is proposed. The proposed IDS effectively captures the temporal and spatial features of CAN messages, leveraging an input generator that exploits

the sequential patterns and local correlations within the messages. Experimental results demonstrate that the proposed IDS significantly reduces memory usage and detection latency while maintaining high detection rates. Specifically, it operates four times faster and utilizes only 3.3% of the memory space required by a full-precision CNN-based IDS. Moreover, it achieves 90.2% of the accuracy of the CNN-based IDS and improves the accuracy of the state-of-the-art BNN-based IDS design by 11.9%. By significantly reducing memory requirements and detection latency while maintaining high accuracy, the proposed IDS can be a promising solution for enhancing the security of in-vehicle networks.

Additionally, we review the literature on rule-based and machine learning-based IDSs for CAN. As there is no benchmark CAN message datasets for this research area, we collect multiple datasets from seven different unmodified licensed vehicles to support future research in this field. All schemes proposed in this research are also thoroughly evaluated using data collected from these real vehicles.

8.2 Future Work

We plan to implement and evaluate our hybrid IDS framework on ECU devices for real-world performance evaluation. This flexible framework allows updates and component replacements based on evolving rules or machine learning algorithms, enabling continuous performance enhancement across diverse environments. Therefore, we also plan to investigate new rules and leverage advancements in machine learning technology to improve performance.

Moreover, we will explore alternative improvement strategies for our BNN-based and BCNN-based IDSs that can enhance performance while maintaining the low latency benefits. One focus is understanding the challenges posed by sophisticated attacks and designing an IDS that offers robust defense against such attacks. In addition, incorporating more complex and varied attack strategies, including attacks generated by generative adversarial networks, will provide a more thorough assessment of the effectiveness of our IDSs.

Our future research also considers using additional CAN data features, such as semantic features, that require collaboration with manufacturers to improve the performance of our IDSs against sophisticated attacks. Overall, our goal is to enhance the performance of IDS systems while ensuring they remain efficient and adaptable to changing threats.

BIBLIOGRAPHY

[1] Kamel Abdelouahab, Maxime Pelcat, Jocelyn Serot, and François Berry. Accelerating cnn inference on fpgas: A survey. *arXiv preprint arXiv:1806.01683*, 2018.

[2] Yong Shen Aiguo Zhou, Zhenyu Li. Anomaly detection of can bus messages using a deep neural network for autonomous vehicles. *Applied Sciences*, 9:3174, 08 2019.

[3] Tom Bannink, Adam Hillier, Lukas Geiger, Tim de Bruin, Leon Overweel, Jelmer Neeven, and Koen Helwegen. Larq compute engine: Design, benchmark and deploy state-of-the-art binarized neural networks. *Proceedings of Machine Learning and Systems*, 3:680–695, 2021.

[4] Josef Berwanger, Christian Ebner, Anton Schedl, Ralf Belschner, Sven Fluhrer, Peter Lohrmann, Emmerich Fuchs, Dietmar Millinger, Michael Sprachmann, Florian Bogenberger, et al. Flexray–the communication system for advanced automotive control systems. *SAE transactions*, pages 303–314, 2001.

[5] Joseph Bethge, Haojin Yang, Marvin Bornstein, and Christoph Meinel. Back to simplicity: How to train accurate bnns from scratch? *arXiv preprint arXiv:1906.08637*, 2019.

[6] Simone Bianco, Remi Cadene, Luigi Celona, and Paolo Napoletano. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277, 2018.

[7] Deborah H Blevins, Pablo Moriano, Robert A Bridges, Miki E Verma, Michael D Iannacone, and Samuel C Hollifield. Time-based can intrusion detection benchmark. In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec), 2021.*, 2021.

[8] Michaela Blott, Thomas B Preußer, Nicholas J Fraser, Giulio Gambardella, Kenneth O'brien, Yaman Umuroglu, Miriam Leeser, and Kees Vissers. Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 11(3):1–23, 2018.

[9] P Borazjani, C Everett, and Damon McCoy. Octane: An extensible open source car security testbed. In *Proceedings of the Embedded Security in Cars Conference*, volume 40, 2014.

[10] Bobba Brao and Kailasam Swathi. Fast knn classifiers for network intrusion detection system. *Indian Journal of Science and Technology*, 10:1–10, 04 2017.

[11] Jason Brownlee. *Probability for Machine Learning: Discover How To Harness Uncertainty With Python*. Machine Learning Mastery, 2019.

[12] Adrian Bulat and Georgios Tzimiropoulos. Binarized convolutional landmark localizers for human pose estimation and face alignment with limited resources. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3706–3714, 2017.

[13] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno, et al. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX security symposium*, volume 4, page 2021. San Francisco, 2011.

[14] Kyong-Tak Cho and Kang G Shin. Fingerprinting electronic control units for vehicle intrusion detection. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 911–927, 2016.

[15] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[16] Steve Corrigan. Introduction to the controller area network (can). *Texas Instrument, Application Report*, 2008.

[17] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[18] Robert I Davis, Alan Burns, Reinder J Bril, and Johan J Lukkien. Controller area network (can) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35:239–272, 2007.

[19] Araya Kibrom Desta, Shuji Ohira, Ismail Arai, and Kazutoshi Fujikawa. Rec-cnn: In-vehicle networks intrusion detection using convolutional neural networks trained on recurrence plots. *Vehicular Communications*, 35:100470, 2022.

[20] Marco Di Natale, Haibo Zeng, Paolo Giusto, and Arkadeb Ghosal. *Understanding and using the controller area network communication protocol: theory and practice*. Springer Science & Business Media, 2012.

[21] Zeinab El-Rewini, Karthikeyan Sadatsharan, Daisy Flora Selvaraj, Siby Jose Plathottam, and Prakash Ranganathan. Cybersecurity challenges in vehicular communications. *Vehicular Communications*, 23:100214, 2020.

[22] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from imbalanced data sets*, volume 11. Springer, 2018.

[23] Sunanda Gamage and Jagath Samarabandu. Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, 169:102767, 2020.

[24] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[25] Mabrouka Gmiden, Mohamed Hedi Gmiden, and Hafedh Trabelsi. An intrusion detection method for securing in-vehicle can bus. In *2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pages 176–180. IEEE, 2016.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[27] Bogdan Groza and Pal-Stefan Murvay. Broadcast authentication in a low speed controller area network. In *International Conference on E-Business and Telecommunications*, pages 330–344. Springer, 2011.

[28] Bogdan Groza, Stefan Murvay, Anthony Van Herrewege, and Ingrid Verbauwhede. Libracan: A lightweight broadcast authentication protocol for controller area networks. In *Cryptology and Network Security: 11th International Conference, CANS 2012, Darmstadt, Germany, December 12-14, 2012. Proceedings 11*, pages 185–200. Springer, 2012.

[29] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

[30] Kaiyuan Guo, Lingzhi Sui, Jiantao Qiu, Jincheng Yu, Junbin Wang, Song Yao, Song Han, Yu Wang, and Huazhong Yang. Angel-eye: A complete design flow for mapping cnn onto embedded fpga. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):35–47, 2017.

[31] Kaiyuan Guo, Shulin Zeng, Jincheng Yu, Yu Wang, and Huazhong Yang. A survey of fpga-based neural network accelerator. *arXiv preprint arXiv:1712.08934*, 2017.

[32] Kyu Suk Han, Swapna Divya Potluri, and Kang G Shin. Real-time frame authentication using id anonymization in automotive networks, March 15 2016. US Patent 9,288,048.

[33] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[34] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[35] Cong Hao, Xiaofan Zhang, Yuhong Li, Sitao Huang, Jinjun Xiong, Kyle Rupnow, Wen-mei Hwu, and Deming Chen. Fpga/dnn co-design: An efficient design methodology for 1ot intelligence on the edge. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2019.

[36] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[37] Haibo He and Yunqian Ma. *Imbalanced learning: foundations, algorithms, and applications*. Wiley-IEEE Press, 2013.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[40] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

[41] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Security threats to automotive can networks–practical examples and selected short-term countermeasures. In *Computer Safety, Reliability, and Security: 27th International Conference, SAFECOMP 2008 Newcastle upon Tyne, UK, September 22-25, 2008 Proceedings 27*, pages 235–248. Springer, 2008.

[42] Md Delwar Hossain, Hiroyuki Inoue, Hideya Ochiai, Doudou Fall, and Youki Kadobayashi. Long short-term memory-based intrusion detection system for in-vehicle controller area network bus. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 10–17. IEEE, 2020.

[43] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *Advances in neural information processing systems*, 29, 2016.

[44] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

[45] Yoshua Bengio Ian Goodfellow and Aaron Courville. *Deep Learning Book*. MIT Press, 2016.

[46] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[47] Poulmanogo Illy, Georges Kaddoum, Christian Miranda Moreira, Kuljeet Kaur, and Sahil Garg. Securing fog-to-things environment using intrusion detection system based on ensemble learning. In *2019 IEEE wireless communications and networking conference (WCNC)*, pages 1–7. IEEE, 2019.

[48] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017.

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

[50] Jin Kim, Nara Shin, Seung Yeon Jo, and Sang Hyun Kim. Method of intrusion detection using deep neural network. In *2017 IEEE international conference on big data and smart computing (BigComp)*, pages 313–316. IEEE, 2017.

[51] Kyounggon Kim, Jun Seok Kim, Seonghoon Jeong, Jo-Hee Park, and Huy Kang Kim. Cybersecurity for autonomous vehicles: Review of attacks and defense. *Computers & Security*, page 102150, 2021.

[52] Shiho Kim, Rakesh Shrestha, Shiho Kim, and Rakesh Shrestha. Introduction to automotive cybersecurity. *Automotive Cyber Security: Introduction, Challenges, and Standardization*, pages 1–13, 2020.

[53] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[56] Manish Kumar, M Hanumanthappa, and TV Suresh Kumar. Intrusion detection system using decision tree algorithm. In *2012 IEEE 14th international conference on communication technology*, pages 629–634. IEEE, 2012.

[57] Takuya Kuwahara, Yukino Baba, Hisashi Kashima, Takeshi Kishikawa, Junichi Tsurumi, Tomoyuki Haga, Yoshihiro Ujiie, Takamitsu Sasaki, and Hideki Matsushima. Supervised and unsupervised intrusion detection based on can message frequencies for in-vehicle network. *Journal of Information Processing*, 26:306–313, 2018.

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

[59] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[60] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[61] Zhengjie Li, Yufan Zhang, Jian Wang, and Jinmei Lai. A survey of fpga design for ai era. *Journal of Semiconductors*, 41(2):021402, 2020.

[62] Shuang Liang, Shouyi Yin, Leibo Liu, Wayne Luk, and Shaojun Wei. Fp-bnn: Binarized neural network on fpga. *Neurocomputing*, 275:1072–1086, 2018.

[63] Chung-Wei Lin and Alberto Sangiovanni-Vincentelli. Cyber-security for the controller area network (can) communication protocol. In *2012 International Conference on Cyber Security*, pages 1–7. IEEE, 2012.

[64] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. *Advances in neural information processing systems*, 30, 2017.

[65] Congli Ling and Dongqin Feng. An algorithm for detection of malicious messages on can buses. In *2012 National Conference on Information Technology and Computer Science*, pages 627–630. Atlantis Press, 2012.

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

[67] Wei Lo, Hamed Alqahtani, Kutub Thakur, Ahmad Almadhor, Subhash Chander, and Gulshan Kumar. A hybrid deep learning based intrusion detection system using spatial-temporal representation of in-vehicle network traffic. *Vehicular Communications*, 35:100471, 2022.

[68] Mirco Marchetti and Dario Stabili. Anomaly detection of can bus messages through analysis of id sequences. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1577–1583. IEEE, 2017.

[69] Mirco Marchetti, Dario Stabili, Alessandro Guido, and Michele Colajanni. Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–6. IEEE, 2016.

[70] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. *arXiv preprint arXiv:2003.11535*, 2020.

[71] Charlie Miller and Chris Valasek. Adventures in automotive networks and control units. *Def Con*, 21(260-264):15–31, 2013.

[72] Charlie Miller and Chris Valasek. A survey of remote automotive attack surfaces. *black hat USA*, 2014:94, 2014.

[73] Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015(S 91), 2015.

[74] Michael R Moore, Robert A Bridges, Frank L Combs, Michael S Starr, and Stacy J Prowell. Modeling inter-signal arrival times for accurate detection of can bus signal injection attacks: a data-driven approach to in-vehicle intrusion detection. In *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*, pages 1–4, 2017.

[75] Duncan JM Moss, Eriko Nurvitadhi, Jaewoong Sim, Asit Mishra, Debbie Marr, Suchit Subhaschandra, and Philip HW Leong. High performance binary neural networks on the xeon+ fpga™ platform. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4. IEEE, 2017.

[76] Michael Müter and Naim Asaj. Entropy-based anomaly detection for in-vehicle networks. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 1110–1115. IEEE, 2011.

[77] Michael Müter, André Groll, and Felix C Freiling. A structured approach to anomaly detection for in-vehicle networks. In *2010 Sixth International Conference on Information Assurance and Security*, pages 92–98. IEEE, 2010.

[78] Sandeep Nair Narayanan, Sudip Mittal, and Anupam Joshi. Using data analytics to detect anomalous states in vehicles. *arXiv preprint arXiv:1512.08048*, 2015.

[79] Sandeep Nair Narayanan, Sudip Mittal, and Anupam Joshi. Obd_securealert: An anomaly detection system for vehicles. In *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–6. IEEE, 2016.

[80] NHTSA. Traffic safety facts annual report. 2022.

[81] Sen Nie, Ling Liu, and Yuefeng Du. Free-fall: Hacking tesla from wireless to can bus. *Briefing, Black Hat USA*, 25:1–16, 2017.

[82] Sen Nie, Ling Liu, Yuefeng Du, and Wenkai Zhang. Over-the-air: How we remotely compromised the gateway, bcm, and autopilot ecus of tesla cars. *Briefing, Black Hat USA*, 2018.

[83] Dennis K Nilsson, Ulf E Larson, and Erland Jonsson. Efficient in-vehicle delayed data authentication based on compound message authentication codes. In *2008 IEEE 68th Vehicular Technology Conference*, pages 1–5. IEEE, 2008.

[84] Nuno Oliveira, Isabel Praça, Eva Maia, and Orlando Sousa. Intelligent cyber attack detection and classification for network-based intrusion detection systems. *Applied Sciences*, 11(4):1674, 2021.

[85] Cornel Reuber Oliver Hartkopp and Roland Schilling. Macan - message authenticated can. In *Escar Conference, Berlin, Germany*, 2012.

[86] Habeeb Olufowobi and Gedare Bloom. Connected cars: Automotive cybersecurity and privacy for smart cities. In *Smart cities cybersecurity and privacy*, pages 227–240. Elsevier, 2019.

[87] Habeeb Olufowobi, Uchenna Ezeobi, Eric Muhati, Gaylon Robinson, Clinton Young, Joseph Zambreno, and Gedare Bloom. Anomaly detection approach using adaptive cumulative sum algorithm for controller area network. In *Proceedings of the ACM Workshop on Automotive Cybersecurity*, pages 25–30, 2019.

[88] Inseok Park and Myoungho Sunwoo. Flexray network parameter optimization method for automotive applications. *IEEE Transactions on Industrial Electronics*, 58(4):1449–1459, 2010.

[89] Kai Peng, Victor CM Leung, and Qingjia Huang. Clustering approach based on mini batch kmeans for intrusion detection system over big data. *IEEE Access*, 6:11897–11906, 2018.

[90] Mert Dieter Pese. *Bringing Practical Security to Vehicles*. PhD thesis, 2022.

[91] Adrien Prost-Boucle, Alban Bourge, Frédéric Pétrot, Hande Alemdar, Nicholas Caldwell, and Vincent Leroy. Scalable high-performance architecture for convolutional ternary neural networks on fpga. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–7. IEEE, 2017.

[92] Andrew Putnam. Fpgas in the datacenter: Combining the worlds of hardware and software development. In *Proceedings of the on Great Lakes Symposium on VLSI 2017*, pages 5–5, 2017.

[93] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognition*, 105:107281, 2020.

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

[95] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[96] Hendrik Schweppe, Yves Roudier, Benjamin Weyl, Ludovic Apvrille, and Dirk Scheuermann. Car2x communication: securing the last meter-a cost-effective approach for ensuring trust in car2x applications using in-vehicle symmetric cryptography. In *2011 IEEE Vehicular Technology Conference (VTC Fall)*, pages 1–5. IEEE, 2011.

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

[98] Alok Sharma, Edwin Vans, Daichi Shigemizu, Keith A Boroevich, and Tatsuhiko Tsunoda. Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture. *Scientific reports*, 9(1):1–7, 2019.

[99] Pramila P Shinde and Seema Shah. A review of machine learning and deep learning applications. In *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, pages 1–6. IEEE, 2018.

[100] Kevin Siu, Dylan Malone Stuart, Mostafa Mahmoud, and Andreas Moshovos. Memory requirements for convolutional neural network hardware accelerators. In *2018 IEEE International Symposium on Workload Characterization (IISWC)*, pages 111–121. IEEE, 2018.

[101] Hyun Min Song, Ha Rang Kim, and Huy Kang 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)*, pages 63–68. IEEE, 2016.

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

[103] Dario Stabili, Mirco Marchetti, and Michele Colajanni. Detecting attacks to internal vehicle networks through hamming distance. In *2017 AEIT International Annual Conference*, pages 1–6. IEEE, 2017.

[104] Xiaoyu Sun, Shihui Yin, Xiaochen Peng, Rui Liu, Jae-sun Seo, and Shimeng Yu. Xnor-rram: A scalable and parallel resistive synaptic architecture for binary neural networks. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1423–1428. IEEE, 2018.

[105] Chris Szilagyi and Philip Koopman. Low cost multicast authentication via validity voting in time-triggered embedded control networks. In *Proceedings of the 5th Workshop on Embedded Systems Security*, pages 1–10, 2010.

[106] Christopher Szilagyi and Philip Koopman. Flexible multicast authentication for time-triggered embedded control network applications. In *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, pages 165–174. IEEE, 2009.

[107] Craig P Szydlowski. Can specification 2.0: Protocol and implementations. Technical report, SAE Technical Paper, 1992.

[108] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *Thirty-First AAAI conference on artificial intelligence*, 2017.

[109] Peiying Tao, Zhe Sun, and Zhixin Sun. An improved intrusion detection algorithm based on ga and svm. *Ieee Access*, 6:13624–13631, 2018.

[110] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. IEEE, 2009.

[111] Adrian Taylor, Nathalie Japkowicz, and Sylvain Leblanc. Frequency-based anomaly detection for the automotive can bus. In *2015 World Congress on Industrial Control Systems Security (WCICSS)*, pages 45–49. IEEE, 2015.

[112] Adrian Taylor, Sylvain Leblanc, and Nathalie Japkowicz. Anomaly detection in automobile control network data with long short-term memory networks. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 130–139. IEEE, 2016.

[113] Daxin Tian, Yuzhou Li, Yunpeng Wang, Xuting Duan, Congyu Wang, Wenyang Wang, Rong Hui, and Peng Guo. An intrusion detection system based on machine learning for can-bus. In *Industrial Networks and Intelligent Systems: 3rd International Conference, INISCOM 2017, Ho Chi Minh City, Vietnam, September 4, 2017, Proceedings 3*, pages 285–294. Springer, 2018.

[114] Miaoqing Tian, Ruobing Jiang, Chaoqun Xing, Haipeng Qu, Qian Lu, and Xiaoyun Zhou. Exploiting temperature-varied ecu fingerprints for source identification in in-vehicle network intrusion detection. In *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8. IEEE, 2019.

[115] Hiroshi Ueda, Ryo Kurachi, Hiroaki Takada, Tomohiro Mizutani, Masayuki Inoue, and Satoshi Horihata. Security authentication system for in-vehicle network. *SEI technical review*, 81:5–9, 2015.

[116] Yaman Umuroglu, Nicholas J Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 65–74, 2017.

[117] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[118] Anthony Van Herrewege, Dave Singelee, and Ingrid Verbauwhede. Canauth-a simple, backward compatible broadcast authentication protocol for can bus. In *ECRYPT Workshop on Lightweight Cryptography*, volume 2011, page 20, 2011.

[119] Stylianos I Venieris and Christos-Savvas Bouganis. fpgaconvnet: A framework for mapping convolutional neural networks on fpgas. In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 40–47. IEEE, 2016.

[120] Stylianos I Venieris, Alexandros Kouris, and Christos-Savvas Bouganis. Toolflows for mapping convolutional neural networks on fpgas: A survey and future directions. *arXiv preprint arXiv:1803.05900*, 2018.

[121] Ravi Vinayakumar, Mamoun Alazab, KP Soman, Prabaharan Poornachandran, Ameer Al-Nemrat, and Sitalakshmi Venkatraman. Deep learning approach for intelligent intrusion detection system. *Ieee Access*, 7:41525–41550, 2019.

[122] Qian Wang, Zhaojun Lu, and Gang Qu. An entropy analysis based intrusion detection system for controller area network in vehicles. In *2018 31st IEEE International System-on-Chip Conference (SOCC)*, pages 90–95. IEEE, 2018.

[123] Wei Wang, Yiqiang Sheng, Jinlin Wang, Xuewen Zeng, Xiaozhou Ye, Yongzhong Huang, and Ming Zhu. Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE access*, 6:1792–1806, 2017.

[124] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. Malware traffic classification using convolutional neural network for representation learning. In *2017 International Conference on Information Networking (ICOIN)*, pages 712–717. IEEE, 2017.

[125] Armin Wasicek, Mert D Pesé, André Weimerskirch, Yelizaveta Burakova, and Karan Singh. Context-aware intrusion detection in automotive control systems. In *Proc. 5th ESCAR USA Conf*, pages 21–22, 2017.

[126] Zhuo Wei, Yanjiang Yang, Yasmin Rehana, Yongdong Wu, Jian Weng, and Robert H Deng. Iovshield: an efficient vehicular intrusion detection system for self-driving (short paper). In *Information Security Practice and Experience: 13th International Conference, ISPEC 2017, Melbourne, VIC, Australia, December 13–15, 2017, Proceedings 13*, pages 638–647. Springer, 2017.

[127] Kehe Wu, Zuge Chen, and Wei Li. A novel intrusion detection model for a massive network using convolutional neural networks. *Ieee Access*, 6:50850–50859, 2018.

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

[129] Yihan Xiao, Cheng Xing, Taining Zhang, and Zhongkai Zhao. An intrusion detection model based on feature reduction and convolutional neural networks. *IEEE Access*, 7:42210–42219, 2019.

[130] Guoqi Xie, Laurence T Yang, Yuanda Yang, Haibo Luo, Renfa Li, and Mamoun Alazab. Threat analysis for automotive can networks: A gan model-based intrusion detection technique. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[131] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, 12(1):40–48, 2010.

[132] Xin Xu. Adaptive intrusion detection based on machine learning: feature extraction, classifier construction and sequential pattern prediction. *international journal of web services practices*, 2(1-2):49–58, 2006.

[133] Tien-Ju Yang, Yu-Hsin Chen, Joel Emer, and Vivienne Sze. A method to estimate the energy consumption of deep neural networks. In *2017 51st asilomar conference on signals, systems, and computers*, pages 1916–1920. IEEE, 2017.

[134] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access*, 5:21954–21961, 2017.

[135] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays*, pages 161–170, 2015.

[136] Linxi Zhang and Di Ma. A hybrid approach toward efficient and accurate intrusion detection for in-vehicle networks. *IEEE Access*, 10:10852–10866, 2022.

[137] Linxi Zhang, Lyndon Shi, Nevrus Kaja, and D Ma. A two-stage deep learning approach for can intrusion detection. In *Proc. Ground Vehicle Syst. Eng. Technol. Symp.(GVSETS)*, pages 1–11, 2018.

[138] Linxi Zhang, Xuke Yan, and Di Ma. Accelerating in-vehicle network intrusion detection system using binarized neural network. *SAE International Journal of Advances and Current Practices in Mobility*, 4(2022-01-0156):2037–2050, 2022.

[139] Linxi Zhang, Xuke Yan, and Di Ma. A binarized neural network approach to accelerate in-vehicle network intrusion detection. *IEEE Access*, 10:123505–123520, 2022.

[140] Shiliang Zhang, Hui Jiang, Shifu Xiong, Si Wei, and Li-Rong Dai. Compact feedforward sequential memory networks for large vocabulary continuous speech recognition. In *Interspeech*, pages 3389–3393, 2016.

[141] Ritchie Zhao, Weinan Song, Wentao Zhang, Tianwei Xing, Jeng-Hau Lin, Mani Srivastava, Rajesh Gupta, and Zhiru Zhang. Accelerating binarized convolutional neural networks with software-programmable fpgas. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 15–24, 2017.