UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUIGI FERNANDO MARQUES DA LUZ

Enhancing Cybersecurity of Automotive Ethernet Networks with Deep Learning-based Intrusion Detection Systems

Recife

2024

LUIGI FERNANDO MARQUES DA LUZ

Enhancing Cybersecurity of Automotive Ethernet Networks with Deep Learning-based Intrusion Detection Systems

M.Sc. Dissertation presented to the Centro de Informática of the Universidade Federal de Pernambuco in partial fulfillment of the requirements for the degree of Master of Computer Science.

Concentration Area: Computer Networks and Distributed Systems

Advisor: Divanilson Rodrigo de Sousa Campelo

Co-advisor: Paulo Freitas de Araujo-Filho

Recife

2024

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Luz, Luigi Fernando Marques da.
Enhancing Cybersecurity of Automotive Ethernet Networks with Deep Learning-based Intrusion Detection Systems / Luigi Fernando Marques da Luz. - Recife, 2024.
102f.: il.

Dissertação (Mestrado) - Universidade Federal de Pernambuco, Centro de Informática, Pós-graduação em Ciência da Computação, 2024.
Orientação: Divanilson Rodrigo de Sousa Campelo.
Coorientação: Paulo Freitas de Araujo-Filho.
Inclui referências.

1. Sistemas de Detecção de Intrusão; 2. Aprendizagem Profunda; 3. Ethernet automotiva. I. Campelo, Divanilson Rodrigo de Sousa. II. Araujo-Filho, Paulo Freitas de. III. Título.

UFPE-Biblioteca Central

UNIVERSIDADE FEDERAL DE PERNAMBUCO

Ata da defesa/apresentação do Trabalho de Conclusão de Curso de Mestrado do Programa de Pós-graduação em Ciências da Computação - CIN da Universidade Federal de Pernambuco, no dia 26 de setembro de 2024.

ATA Nº 2162

Ao vigésimo sexto dia do mês de setembro do ano de dois mil e vinte e quatro, às catorze horas, no Centro de Informática da Universidade Federal de Pernambuco, teve início a duas milésima centésima sexagésima segunda defesa de dissertação do Mestrado em Ciência da Computação, intitulada Enhancing Cybersecurity of Automotive Ethernet Networks with Deep Learning-based Intrusion Detection Systems, na área de concentração de Redes de Computadores e Sistemas Distribuídos, do candidato Luigi Fernando Marques da Luz o qual já havia preenchido anteriormente as demais condições exigidas para a obtenção do grau de mestre. A Banca Examinadora, composta pelos professores Daniel Carvalho da Cunha, pertencente ao Centro de Informática desta Universidade, Lourenço Alves Pereira Junior, pertencente ao Departamento de Sistemas de Computação do Instituto Tecnológico de Aeronáutica e Divanilson Rodrigo de Sousa Campelo, pertencente ao Centro de Informática desta Universidade, sendo o primeiro presidente da banca examinadora e o último orientador do trabalho de dissertação, decidiu: Aprovar o trabalho. E para constar lavrei a presente ata que vai por mim assinada e pela Banca Examinadora. Recife, 26 de setembro de 2024.

Dr. LOURENÇO ALVES PEREIRA JUNIOR, ITA

Examinador Externo à Instituição

Dr. DIVANILSON RODRIGO DE SOUSA CAMPELO, UFPE

Examinador Interno

Dr. DANIEL CARVALHO DA CUNHA, UFPE

Presidente

LUIGI FERNANDO MARQUES DA LUZ

Mestrando(a)

To my beloved grandparents, Fernando and Adelaide, who unfortunately are not here anymore to share this moment with me, but I am sure they would be as cheerful as they always were in every other moment of my life.

ACKNOWLEDGMENTS

To my mother, Flavia, for saying she is proud of me and loves me many times daily. Your kind words, hugs, and kisses helped me get through the difficult moments, and knowing that I always had someone to rely on made every step and decision I took in my life easier.

To my father, Luiz, who supported me in his way during my entire life.

To my advisors, Divanilson Campelo and Paulo Freitas, who actively guided and supported me through the field of scientific research, sharing valuable knowledge and life experiences that helped me improve as a researcher and a person.

To my former advisor, Fernando Gonçalves, who accepted me as an undergraduate researcher in my second year of college and guided me throughout my entire graduation, these experiences played a huge part in my decision to pursue a master's degree.

To my friends, Cristóvão, Eron, Manu, Natasha, Ricardo, Victor Cabral, Vitor Barros, and Vitor Coutinho, who were there for me in the ups and downs, cheering me up with jokes, chatting, listening to my complaints, and sharing joyful moments making barbecues and eating burgers.

To the examining board professors, Daniel da Cunha and Lourenço Júnior, for the valuable comments that helped improve the quality of this dissertation.

To every other person, I believe the interactions and experiences we have shared in the past years are part of who I am today.

ABSTRACT

Modern automobiles are increasing the demand for automotive Ethernet as a flexible and high-bandwidth in-vehicle network (IVN) technology. Moreover, while enhanced connectivity brings new opportunities and capabilities to cars, it also presents security concerns to drivers and passengers. Traditional network security mechanisms (e.g., encryption and authentication) have drawbacks, such as computing and transmission overhead, when considered for IVNs. On the other hand, intrusion detection systems (IDSs) are a second line of defense triggered when other security mechanisms fail. Alongside, IDSs have smaller deployment costs and do not require modification of existing nodes' message structures. Meanwhile, machine learning (ML) and deep learning (DL) techniques have shown promising results for designing IDSs because of their ability to learn hidden patterns in complex data, such as the network packets in IVNs. However, DL models often demand high computational power and storage size, making their adoption difficult in resource-constrained environments such as IVNs. In this dissertation, we propose two DL-based IDSs that target a low detection time and accurate cyberattack detection. Our first proposal is a deep learning-based intrusion detection system for detecting cyberattacks in an automotive Ethernet network. It uses a convolutional neural network architecture and a multi-criteria optimization technique. Our experimental results show a reduction of more than 100x in the storage size and a speedup of 1.4x in the detection time with a negligible drop in the F1-score compared to state-of-the-art work. The second proposal is a novel multi-stage deep learning-based intrusion detection system to detect and classify cyberattacks in automotive Ethernet networks. The first stage uses a Random Forest classifier to detect cyberattacks quickly. The second stage, on the other hand, uses a Pruned Convolutional Neural Network that minimizes false positive rates while classifying different types of cyberattacks. We evaluate our proposed IDS using two publicly available automotive Ethernet intrusion datasets. The experimental results show that our proposed solution detects cyberattacks with a similar detection rate and a faster detection time compared to other state-of-the-art baseline automotive Ethernet IDSs.

Key-words: intrusion detection system; deep learning; automotive Ethernet.

RESUMO

Automóveis modernos tem aumentado sua demanda por Ethernet automotiva como uma tecnologia flexível e de alta largura de banda de redes intraveiculares. Além disso, enquanto a conectividade nos veículos traz novas oportunidades e funcionalidades, também apresenta problemas de cibersegurança. Métodos como criptografia e autenticação tem pontos negativos, como sobrecarga de computação e de transmissão, se considerados para redes intraiveculares. Entretanto, sistemas de detecção de intrusão (IDSs, do inglês *Intrusion Detection Systems*) são uma segunda linha de defesa e são ativados quando os outros mecanismos de defesa falham. IDSs possuem um menor custo de implantação e não geram mudanças nas estruturas de mensagens. Enquanto isso, técnicas de aprendizagem de máquina e aprendizagem profunda (DL, do inglês *Deep Learning*) têm apresentado resultados promissores no projeto de IDSs, por suas capacidades de aprender padrões escondidos em dados complexos, como os pacotes de redes intraveiculares. Contudo, modelos de DL costumam demandar um alto poder computacional e espaço de armazenamento, o que dificulta sua utilização em ambientes com limitações computacionais, como as redes intraveiculares. Nesta dissertação, são propostos dois IDSs baseados em DL visando uma rápida e precisa detecção de ataques cibernéticos. Inicialmente, é proposto um IDS baseado em DL para detectar ataques em redes Ethernet automotiva. Este IDS usa uma arquitetura de rede neural convolucional e uma técnica de otimização multicritério. Os resultados experimentais obtidos apresentam uma diminuição de mais de 100 vezes na quantidade de memória necessária para armazenar o modelo e um tempo de detecção 40% mais rápido, em troca de um pequeno decréscimo no F1-Score em comparação com trabalhos existentes. A segunda proposta apresenta um IDS baseado em DL com múltiplos estágios para detecção e classificação de ataques cibernéticos em redes Ethernet automotiva. O primeiro estágio usa um classificador *Random Forest* para detectar atividades maliciosas rapidamente. Já o segundo estágio, usa uma rede neural convolucional podada que minimiza as taxas de falsos positivos e também classifica diferentes tipos de ataques. Esta segunda proposta de IDS foi avaliada em dois datasets públicos para detecção de intrusão em redes Ethernet automotiva. Os resultados experimentais apresentam uma taxa de detecção similar e um tempo de detecção mais rápido em comparação com trabalhos apresentados na literatura.

Palavras-chaves: sistemas de detecção de intrusão; aprendizagem profunda; Ethernet automotiva.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

2D-CNN Two Dimensional Convolutional Neural network

ADAS Advanced Driver Assistance System

AE Auto Encoder

AEID Automotive Ethernet Intrusion Dataset

AH Authentication header

ATT&CK Adversarial Tactics, Techniques and Common Knowledge

AVB Audio/Video Bridging

AVTP Audio Video Transport Protocol

AVTPDU Audio Video Transport Protocol Data Unit

BMCA Best Master Clock Selection Algorithm

CAE Convolutional Auto Encoder

CAM Content Adressable Memory

CAN Controller Area Network

CAN FD Controller Area Network with Flexible Datarate

CNN Convolutional Neural Network

CPU Central Processing Unit

DL Deep Learning

DLC Data Length Code

DoS Denial-of-Service

ECU Electronic Control Unit

EMC ElectroMagnetic Compatibility

ESP Encapsulating Security Payload

FPR False Positive Rate

GAN Generative Adversarial Networks

gPTP generalized-Precision Time Protocol

GPU Graphical Processing Unit

HU Head Unit

IDS Intrusion Detection Systems

IKE Internet Key Exchange

IVN In-Vehicle Networks

LIN Local Interconnect Network

LSTM Long Short Term Memory

MAC Media Access Control

MACsec Media Access Control Security

ML Machine Learning

MOST Media Oriented Systems Transport

OCSVM One-Class Support Vector Machine

OEM Original Equipment Manufacturer

Pruned CNN IDS Pruned Convolutional Neural Network Intrusion Detection System

QoS Quality of Service

RNN Recurrent Neural Network

ROC Receiver Operating Characteristic

ROC AUC Receiver Operating Characteristic Area Under Curve

RSE Rear Seat Entertainment

SDN Software Defined Vehicle

SecOC Security Operations

SOME/IP Scalable Service-Oriented Middleware over IP

SRP Stream Reservation Protocol

SVM Support Vector Machine

TLS Transport Layer Security

TLV Type Length Value

TPR True Positive Rate

TSN Time Sensitive Networking

UDP User Datagram Protocol

UNECE United Nations Economic Comission for Europe

UTSP Unshielded Twisted Single Pair

VLAN Virtual Local Area Networking

XGBoost Extreme Gradient Boosting

TABLE OF CONTENTS

1 INTRODUCTION

Today's cars contain dozens of electronic control units (ECUs), which are interconnected by a number of in-vehicle networks (IVNs), (LAI et al., 2020; WU et al., 2020). Recent trends in cars, such as the distribution of automotive functions among different ECUs, higher bandwidth demands from new sensor types (e.g., cameras), and the paradigm shift to a service-oriented architecture have made legacy IVN technologies like controller area network (CAN) unsuitable for supporting the aforementioned tendencies because of its limited bandwidth and limited scalability for future applications. The emergence of Ethernet as a high-bandwidth and flexible IVN solution, especially after the standardization of the IEEE 100BASE-T1 Ethernet, has opened a myriad of opportunities for the introduction of new technologies in vehicles (MATHEUS; KÖNIGSEDER, 2021).

To provide Ethernet with quality of service (QoS) capabilities, the Audio Video Bridging (AVB)/Time Sensitive Networking (TSN) task groups defined several standards that offer time synchronization, low latency, and reliability in switched Ethernet networks (MATHEUS; KÖNIGSEDER, 2021; TUOHY et al., 2015). For instance, the IEEE 1722-2016 standard defines the audio-video transport protocol (AVTP), which guarantees the reliable transmission of high bandwidth time-sensitive Ethernet traffic such as video frames from automotive infotainment systems (IEEE, 2016). Another example is the generalized precision time protocol (gPTP), which synchronizes the nodes to a common reference time to render the transmitted streams in sync (IEEE, 2020). It is worth noting, however, that Ethernet-based communications in cars must coexist with legacy in-vehicle technologies such as CAN, which is still used for safety-control applications due to its low cost and efficiency.

However, while enhanced connectivity brings new opportunities and capabilities to cars, it also presents security concerns to drivers and passengers (LIU et al., 2017; JO; CHOI, 2021; GHOSAL; CONTI, 2020). (CHECKOWAY et al., 2011) demonstrated the feasibility of remote exploitation of vehicles via connectivity tools, such as Bluetooth and cellular radio. Additionally, it was shown that hijacked wireless communication channels allow long-distance vehicle control and theft. As it follows, (MILLER; VALASEK, 2015) showed the steps to remotely hack a car and turn off its engine on a highway – the vulnerabilities they found resulted in a recall of 1.4 million vehicles. Alongside, (JEONG et al., 2021) demonstrated a replay attack on an automotive Ethernet scenario. This replay attack can manipulate information and mislead the

decision-making process of an autonomous vehicle, posing a significant threat to people's lives. So, defending vehicles against security threats is crucial in today's connected cars.

Traditional network security mechanisms include encryption and authentication. However, some of these mechanisms have drawbacks when considered for a resource-constrained environment such as IVNs. For example, encryption adds computing and transmission overhead that may not be suitable for IVN timing requirements (JO; CHOI, 2021). On the other hand, intrusion detection systems (IDS) are security mechanisms that work as a second line of defense, triggered when other security measures fail. IDSs monitor devices and networks to identify intrusions and report malicious activities. One of the IDS benefits is that it can be deployed as a separate network node, excluding the necessity of modifying existing nodes to add encryption or authentication (WU et al., 2020).

IDSs can be classified as signature and anomaly-based. Signature-based IDSs rely on previous information regarding existing cyberattacks, creating the need for constant updates as new attacks are developed every day (NISIOTI et al., 2018). On the other hand, anomaly-based IDSs identify a normal pattern of the network/system and detect cyberattacks based on their deviation from the normal behavior, overcoming the major drawback of signature-based IDSs but at the cost of a higher false positive rate. Although anomaly-based IDSs can use traditional statistical methods such as interquartile range outlier detection to detect anomalies, machine learning (ML) based IDSs have gained attention because of their detection results and capability of detecting more complex attacks (ARAUJO-FILHO et al., 2020).

Moreover, a modern vehicle produces tons of data representing the vehicle components' behavior, which could be further employed to develop ML-based IVN IDSs (LAI et al., 2020; WU et al., 2020). However, ML-based IDSs usually demand high computational power, often unavailable in IVNs (BIANCO et al., 2018). Additionally, according to (UN Regulation 155, 2021), vehicles manufactured after July 2022 in countries within the United Nations Economic Commission for Europe (UNECE) jurisdiction must be able to detect and report cyberattacks.

1.1 PROBLEM STATEMENT

Despite the previously mentioned benefits of the use of ML/DL-based IDSs for detecting cyberattacks in automotive environments, there are still open challenges that need to be further addressed and are considered for this dissertation:

- While IDSs for IVNs must accurately detect malicious attacks, they still need a low detection time and a small storage size to be deployed in resource-constrained environments. Thus, it is necessary to propose new detection solutions that aim to optimize the detection accuracy, detection time, and storage size simultaneously, targeting IVN environments;
- While IDSs must report malicious activities quickly, it is important to have a classification of such events to provide information for forensics and future improvements (WU et al., 2020). Thus, it is necessary to propose new detection solutions that can accurately detect malicious activity quickly and provide a classification of it.

1.2 RESEARCH OBJECTIVES

The impacts of cyberattacks in the automotive industry can be significant, potentially affecting people's lives. Therefore, our research aims to investigate whether DL-based IDSs can improve the security of automotive Ethernet networks by identifying cyberattacks. We seek to contribute to the advancement of the automotive cybersecurity field by developing new IDS architectures to address these challenges. Our main objective is to improve the security of automotive Ethernet networks by accurately detecting and categorizing malicious activities with a low detection time using DL-based IDSs. To achieve this goal, we have defined the following specific objectives:

1. Propose an IDS to detect malicious traffic in an AVTP network that uses a multi-criteria optimization technique that improves detection results, storage size, and detection time;
2. Propose a novel multi-stage DL-based IDS, in which the first stage goal is to quickly detect cyberattacks, while the second stage aims to detect and classify the cyberattacks with a lower false positive rate.

1.3 PUBLICATIONS GENERATED BY THIS MASTER'S WORK

This section summarizes the author's publications on the intrusion detection systems for automotive Ethernet networks field during his Master's Degree studies:

1. Paper "Multi-criteria optimized deep learning-based intrusion detection system for detecting cyberattacks in automotive Ethernet networks". In: Anais do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. SBC, 2023. p. 197-210. doi: <https://doi.org/10.5753/sbrc.2023.527> (LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023). This paper was selected as one of the best papers of SBRC 2023. We were invited to submit an extended version to a special issue of the Ad Hoc Networks Journal;

2. Paper "Multi-stage deep learning-based intrusion detection system for automotive Ethernet networks". Ad Hoc Networks, v. 162, p. 103548, 2024. doi: <https://doi.org/10.1016/j.adhoc.2024.103548> (LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2024).

1.4 DISSERTATION OVERVIEW

The remainder of this work is described as follows: Chapter 2 depicts the timeline of IVNs, architectures, and protocols, focusing on automotive Ethernet. Chapter 3 covers the need for security mechanisms for in-vehicle networks, focusing on intrusion detection systems. Chapter 4 summarizes the state-of-the-art work in the field of intrusion detection systems for automotive Ethernet, depicting the contributions and drawbacks of the existing work. Chapter 5 describes our first IDS proposal, which uses a multi-criteria optimization technique to improve detection results, storage size, and detection time, describing the threat model, methodology, and results. As it follows, chapter 6 describes our second IDS proposal, which uses a multi-stage approach to obtain a low detection time and maintain accurate detection results, covering the threat model, experimental setup, evaluation, and comparison with other work. Finally, Chapter 7 concludes this dissertation and discusses possible future works related to intrusion detection systems for the automotive networks field.

2 IN-VEHICLE NETWORKS

This chapter covers the evolution of IVN architectures over time. In addition, it describes the communication protocols and technologies used in automotive networks, with a more in-depth discussion of automotive Ethernet.

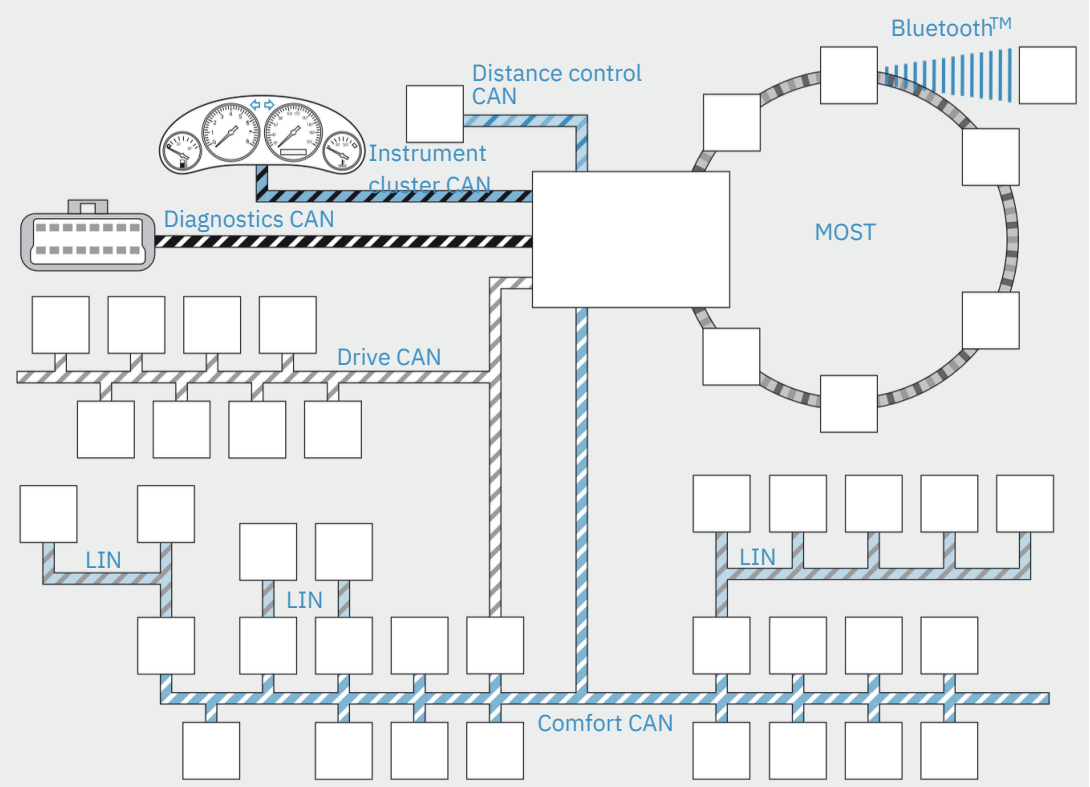
2.1 IN-VEHICLE NETWORKS ARCHITECTURES AND TRENDS

Initially, vehicles were simple mechanical machines with wheels and no windows. The introduction of power windows to improve the user experience brought the need to use electronic sensors, actuators, and ECUs to implement such features. As a natural evolution, some features needed to communicate with each other to exchange information, which was obtained by point-to-point links. However, the growing number of features led to an impractical number of point-to-point links since these links were cables, the third heaviest and most expensive component in a car (MATHEUS; KÖNIGSEDER, 2021). For example, for the driver to control all the windows locally, it was necessary to have a link between the driver door ECU and every other door ECU. If every ECU needed to communicate with each other, $\frac{n(n-1)}{2}$ connections would be necessary, where "n" is the number of ECUs.

To address the issue of the huge number of connections, the CAN protocol was proposed, using a shared communication media, the communication bus. With CAN, all ECUs are connected and able to communicate with each other. Additionally, CAN offers robust and reliable real-time communication between ECUs, with a bandwidth of up to 1 Mbps. However, CAN bandwidth is insufficient for data-demanding applications such as multimedia. In this direction, the automotive industry started incorporating different protocols for different applications, use cases, and requirements, as illustrated in Figure 1. For instance, CAN has been used for critical systems, Media Oriented Systems Transport (MOST) for multimedia and infotainment, and Local Interconnect Network (LIN) for simple applications such as light controls. On the other hand, this multitude of protocols has increased network management and integration complexity. For example, a modern vehicle may have more than 100 ECUs, with an almost one-to-one ECU-feature correspondence (PARET; REBAINE, 2022).

To improve the scalability of IVN architectures, there has been a shift to centralized architectures, mainly based on integrating several related functions onto a single ECU. The

Figure 1 – Luxury vehicle IVN architecture using different protocols.

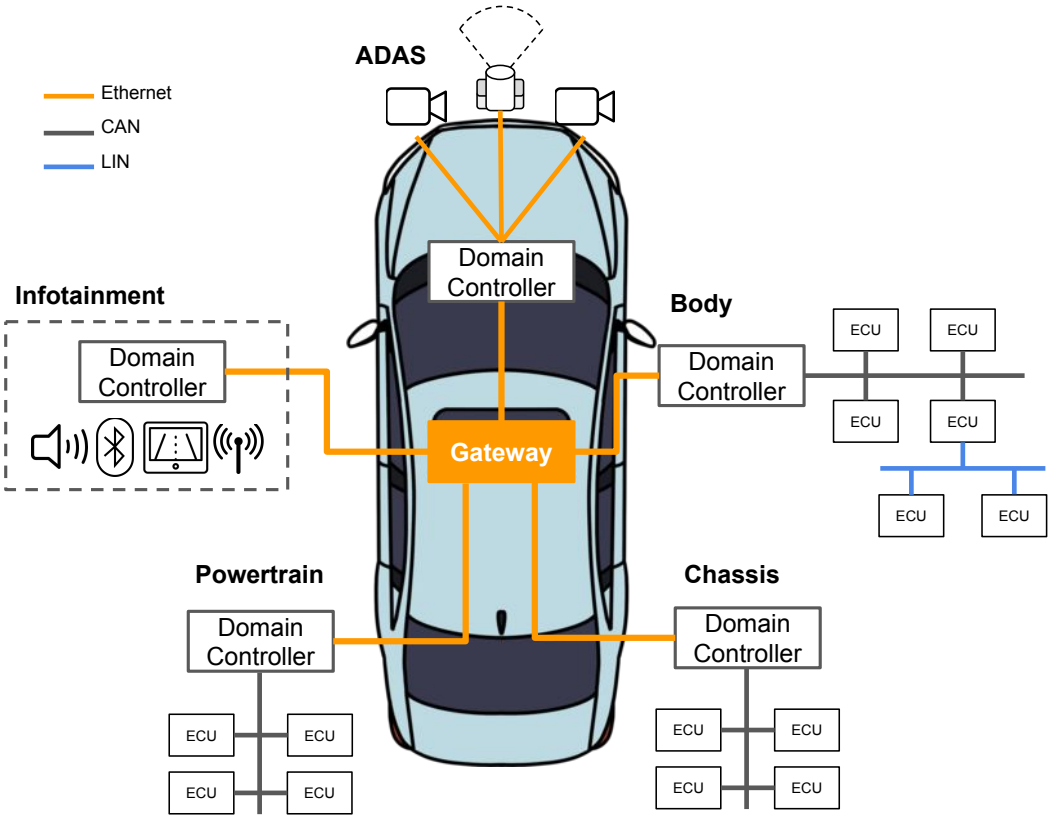


Source: (BOSCH, 2014)

domain-oriented architecture (as shown in Figure 2) is one of the types of centralization that introduces the domain-controller as an entry point and main controller of a specific domain (that can be powertrain, chassis, and so on) and can communicate with other domains through the gateway. Alongside, a domain-based architecture acts as a supporting technology for software-defined vehicles (SDN), as the centralization eases the software updates once the main updates are regarding the domain controller software and the ECUs are simpler devices that captures hardware variations (BANDUR et al., 2021). In this scenario, automotive Ethernet acts as an enabling technology, being the backbone network for the domain controllers, while offering a high bandwidth that supports applications such as autonomous driving and advanced driver assistance systems (ADAS) (BELLO; PATTI; LEONARDI, 2023).

Furthermore, advances in vehicle connectivity have enabled vehicles to share and consume data with the outside world. However, this came with drawbacks, such as the increased attack surface and cyberattack vulnerabilities, bringing the attention to consider security aspects when designing IVNs (GHOSAL; CONTI, 2020). In the following sections, we discuss three of the most common protocols used in a heterogeneous IVN, discussing their background, working principles, and security aspects.

Figure 2 – Domain-based intra-vehicular network architecture.



Source: The author (2024)

2.2 TECHNOLOGIES FOR IVNS

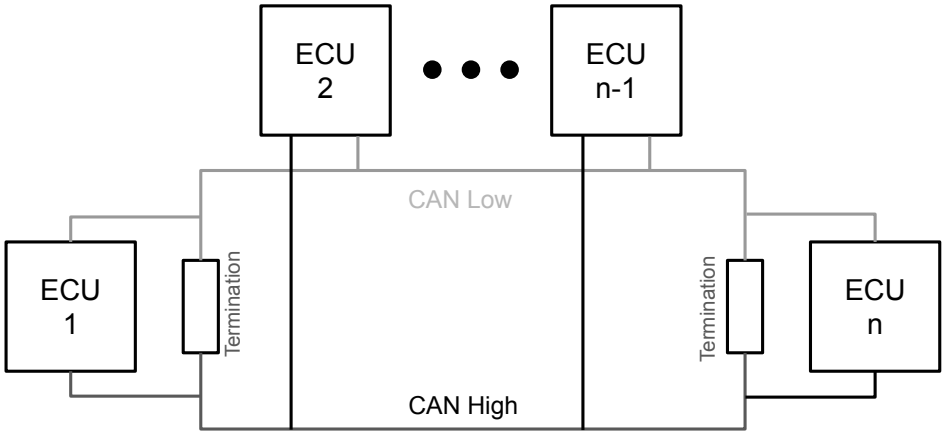
In this section, we discuss the specificities of CAN, gPTP, and AVTP protocols, which are the foundational protocols for this dissertation, covering their working principles, frame structures, and security aspects.

2.2.1 Controller Area Networks

BOSCH introduced the CAN protocol in the 1980s (BOSCH et al., 1991). CAN widespread adoption has been enhanced by several factors, such as open licensing, which allowed other Original Equipment Manufacturers (OEMs) to use the technology, early cooperation with semiconductor companies to produce CAN hardware, and the fact that BOSCH was also a user of its technology. Additionally, CAN has been proven robust and cost-effective and has provided sufficient performance for many applications, including safety-critical systems control in vehicles and other industries. Figure 3 highlights the CAN topology, which uses a bus as

a shared data medium, eliminating the need for a vast number of point-to-point connections between the ECUs and reducing the necessary amount of wires, directly reducing the weight and cost of a vehicle (MATHEUS; KÖNIGSEDER, 2021).

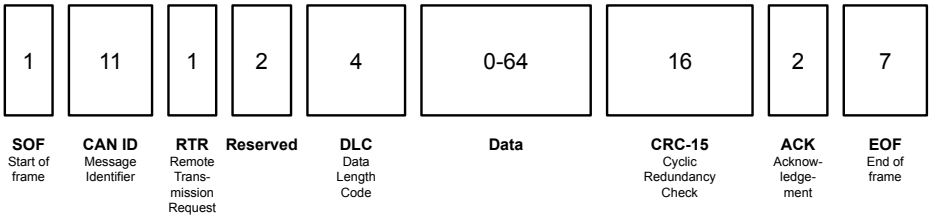
Figure 3 – CAN bus with an arbitrary number of ECUs.



Source: The author (2024)

Each ECU in a CAN bus transmits messages in the CAN frame format presented in Figure 4. The most important fields are the CAN ID, which determines the priority of the message and is used in the arbitration phase to determine which message will be transmitted in the bus; the data length code (DLC), which tells how many bytes of data the payload carries, and the payload itself, which carries the ECU data (BOSCH et al., 1991).

Figure 4 – CAN frame format.

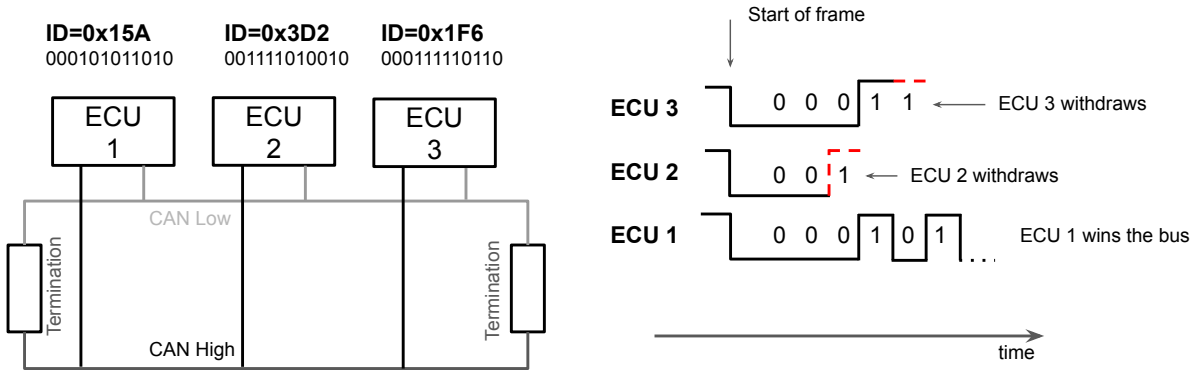


Source: The author (2024)

In a CAN system, arbitration determines which message will win the bus and be transmitted. The arbitration logic dictates that the message with the highest priority will win the bus over messages with lower priority. The priority is defined by the CAN ID, where the lowest ID indicates the highest priority. The CAN bus arbitration is based on electric principles, distinguishing between “dominant” (bit 0) and “recessive” (bit 1) bits in the CAN message ID field. The bus works as a “logical AND” operation: when devices send a 1, the bus remains at

1, but when devices send 0, the bus will be pulled to a 0 value (CORREA et al., 2014). Figure 5 shows an example of the CAN bus arbitration. In this example, three ECUs try to transmit a message simultaneously, beginning the arbitration phase. ECU 1, 2 and 3 have the CAN IDs 0x15A, 0x3D2 and 0x1F6, respectively. During the ID transmission, ECU 2 CAN ID's third bit is recessive, making it withdraw from the arbitration. As follows, the fifth bit of ECU 3 CAN ID is recessive while ECU 1 ID's fifth bit is dominant. Therefore, ECU 1 wins the bus and transmits its message, and ECU 3 withdraws. The ECUs that loses the arbitration keeps its message for later transmission.

Figure 5 – CAN bus arbitration process example with three ECUs.



Source: The author (2024)

The CAN standard has a data rate of up to 1 Mbit/s. However, vehicle applications started to need more bandwidth and sophisticated functionalities. For these reasons, CAN has different generations, some of which are presented in Table 1. CAN with Flexible Datarate (CAN FD) allows a data rate from 5 to 8 Mbit/s when transmitting the payload field (HARTWICH et al., 2012). More recently, BOSCH presented CAN XL, which offers several improvements, including a bit rate of up to 20 Mbit/s in the payload transmission, a larger payload size, functionalities such as virtual CAN networks, Ethernet tunneling, and an acceptance field that supports content and node-based addressing (HARTWICH; BOSCH, 2020).

Table 1 – CAN, CAN FD, and CAN XL comparison.

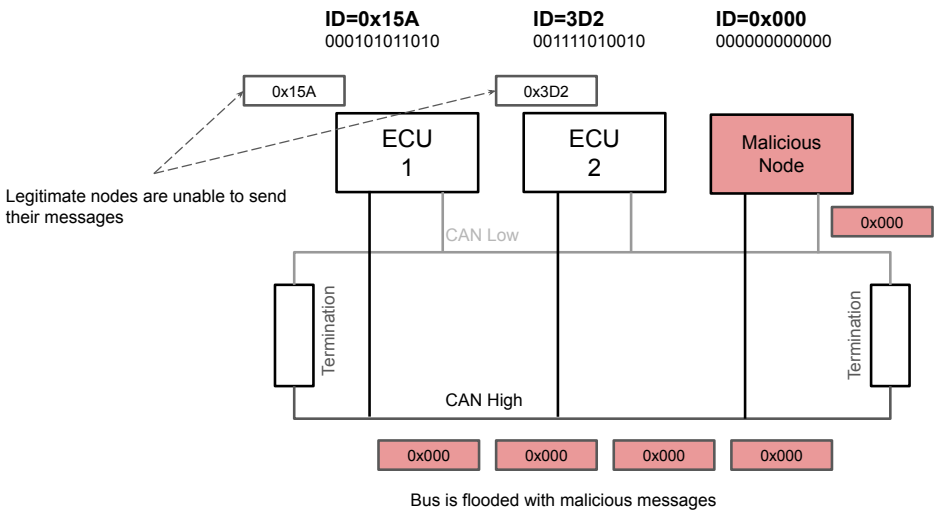
Specification	ID (bits)	Payload (bytes)	Data rate
CAN	11 and 29	0 to 8	up to 1 Mbit/s
CAN FD	11 and 29	0 to 64	payload: from 5 to Mbit/s
CAN XL	11	1 to 2048	payload: up to 20 Mbit/s

Source: The author (2024)

Since IVNs may comprise several protocols, especially in modern vehicles, one approach to enable CAN-Ethernet networks communication (further discussed in Chapter 6) is to tunnel CAN frames within UDP frames, where the CAN ID is inserted in the UDP port number, and the CAN DLC and payload are placed inside the UDP payload (HAN; KWAK; KIM, 2023).

Security aspects. CAN does not support intrinsic security features such as encryption or authentication. As vehicle connectivity increases, some aspects of the CAN protocol make it prone to malicious attacks (JO; CHOI, 2021; LIU et al., 2017). One significant aspect is the CAN bus arbitration process, where the message with the lowest ID wins the bus. In the scenario depicted in Figure 6, if a malicious node has access to the bus, it can carry denial-of-service (DoS) attacks by sending messages with high-priority IDs (e.g., 0x000), which flood the bus and difficult the communication between legitimate nodes. Malicious nodes can also impersonate legitimate nodes and send fake or manipulated messages, which the destination nodes will successfully handle since the standard CAN frame cannot check the host's authenticity (SEO; SONG; KIM, 2018).

Figure 6 – Denial-of-Service attack in a CAN network.



Source: The author (2024)

2.2.2 Automotive Ethernet

The first use case of Ethernet in vehicles was to speed up the time taken to update the software of vehicle's ECUs. In the early 2000s, BMW estimated that by 2008, it would take up

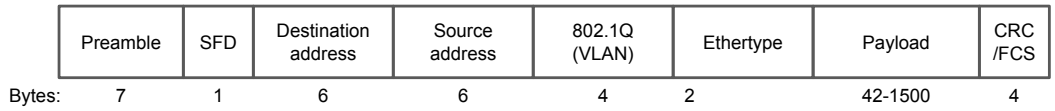
to 16 hours to fully update the software of the vehicles' ECUs using CAN with a practical data rate of 200 kbps. Therefore, BMW set a target software update time of 15 minutes. Technologies such as MOST and USB were evaluated. However, despite MOST providing sufficient bandwidth, its ring topology was not favorable to attaching a node temporarily, and it was a new interface for external testers. Moreover, USB had the disadvantages of expensive cables and connectors and lack of network support. Meanwhile, 100BASE-TX Ethernet provided a sufficient data rate, enabled the car to be handled as a network node inside a more extensive diagnosis network, and was already available on computers. For these reasons, Ethernet was chosen as the most adequate technology for performing ECU software updates (MATHEUS; KÖNIGSEDER, 2021).

The second use case of Ethernet in vehicles was to implement the feature of audio and video data transmission from the Head Unit (HU) to a Rear Seat Entertainment (RSE) at a 20 Mbps rate. Initially, MOST was initially considered for this application. However, its ring topology would increase the complexity of temporarily adding a test node, and it would have been a completely new interface for the external testers. On the other hand, Ethernet provided a sufficient data rate, but it suffered from ElectroMagnetic Compatibility (EMC) from the vehicle components. Under these circumstances, BMW reached Broadcom technology and developed the Unshielded Twisted Single Pair (UTSP) Ethernet, allowing Ethernet PHY with only a pair of cables and not suffering from EMC. The rise of UTSP Ethernet was pivotal to start considering a widespread adoption of Ethernet within vehicles (MATHEUS; KÖNIGSEDER, 2021).

As the vehicles' bandwidth requirements were constantly upgraded, it was impractical to consider changing the IVN technology every time a new feature emerged. This is one of the reasons that favor a wider adoption of Ethernet. Ethernet is flexible, scalable, and compatible with protocols already proof-tested in other industries. For instance, the Ethernet frame (depicted in Figure 7) allows the use of upper-layer protocols based on its Ethertype and wide payload field (CORREA et al., 2014). Other fields, such as the destination and source addresses, provide fundamental addressing mechanisms to ensure the authenticity of messages, and the 802.1Q field enables the implementation of virtual local area networking (VLAN) and specifying a frame priority level. To advocate the introduction of Ethernet on a large scale inside a vehicle, in 2011, the One Pair Ethernet (OPEN) Alliance (initially formed by NXP, Broadcom, and BMW) was created to establish Ethernet as an IVN technology.

Furthermore, despite Ethernet was successfully used in the RSE use case, there were still

Figure 7 – The Ethernet frame.



Source: Adapted from: (CORREA et al., 2014)

challenges, such as QoS, to guarantee timely data delivery, low cost, open standards, and wide supplier choice. Ethernet was mainly based on best-effort communication and did not offer determinism for packet transport. To address the determinism issue for multimedia data transmission, the IEEE AVB task group was formed and developed standards for highly reliable packet networks for latency-constrained applications such as those found in automobiles (BELLO; PATTI; LEONARDI, 2023).

The primary focus of the automotive industry was to use AVB to stream audio and video within the vehicle. The initial standards, known as AVBgen1, defined several protocols, where the Stream Reservation Protocol (SRP) and gPTP (further detailed in Section 2.2.2.1) were considered the foundational protocols of automotive Ethernet (CORREA et al., 2014), as they handled the reservation of time slots for high-priority traffic and the time synchronization between the devices. Additionally, AVBgen1 defined two transport protocols, one responsible for transport in layer 2 (IEEE 1722, which is further described in Section 2.2.2.2) and the other for transport in layer 3 (IEEE 1733). The AVBgen1 protocols are listed below with a brief description of their functionalities:

- IEEE 802.1Qav, "Forwarding and Queuing Enhancements for Time-Sensitive Streams"(traffic shaping), January 5, 2010.
- IEEE 802.1Qat, "Stream Reservation Protocol,"September 30, 2010.
- IEEE 802.1AS, "Timing and Synchronization for Time-Sensitive Applications,"March 30, 2011.
- IEEE 1733, "Protocol for Time-Sensitive Applications in Local Area Networks"(AVB adaption of RTP), April 25, 2011.
- IEEE 1722, "Transport Protocol for Time-Sensitive Applications in a Bridged Local Area Network"(layer two transport protocol), May 6, 2011.

- IEEE 802.1BA, "Audio Video Bridging (AVB) System"(overall system configuration, profiles) September 30, 2011.
- IEEE 1722.1, "Device Discovery, Connection Management, and Control Protocol for 1722 Based Devices"(control mechanisms and service discovery), August 23, 2013.

Once the AVBgen1 set of protocols was published, the next step was to investigate Ethernet for safety-critical data, which could be used in applications such as autonomous driving. Therefore, since the focus was not only on audio/video anymore, the task group was renamed to TSN. Table 2 presents the new set of TSN standards, or AVBgen2, which included improvements in the AVBgen1 standards and introduced new protocols that covered important aspects such as security and redundancy.

Table 2 – AVB and TSN set of protocols.

Set	**Transport**	**Time sync**	**Stream reservation**	**QoS/latency**	**Safety (seamless redundancy)**	**Ingress Policing**
AVB (AVBgen1)	1722-2011	802.1AS-2011	802.1Qat-2010	802.1Qav-2009		
TSN (AVBgen2)	1722-2016	802.1AS-2020	802.Qcc-2018	802.1Qbv-2015	802.1Qca-2015	802.1Qci-2017
				802.1Qbu & 802.3br-2016	802.1CB-2017	
				802.1Qch-2017	802.1AS-2020	
				802.1Qcr-2020 (est.)		

Source: The author (2024)

The IEEE 1722-2016 incorporated more common A/V formats, including encrypted packet formats, UDP/IP encapsulation, and tunneling messages from typical automotive legacy technologies such as CAN, LIN, MOST, and FlexRay (IEEE, 2016). Additionally, IEEE802.1AS-2020 mainly focused on including redundancies in the gPTP specifications to offer the ability for the network to recover as soon as possible in cases where a connection is lost (IEEE, 2020). Regarding the new standards, TSN introduced ingress policing with the IEEE 802.1Qci. This was a significant advance in the security of Ethernet networks, as this protocol defines means to drop frames based on measures of incoming data streams per packet basis, preventing the switch from being inundated with excess erroneous traffic (LOCAL; NETWORKS-BRIDGES; NETWORKS, 2017).

In the following sections, we mainly focus on the IEEE 802.1AS (gPTP) and IEEE 1722 (AVTP), as they are the common baseline for precise synchronization between nodes in au-

dio/video transmission applications in automotive networks (JEONG et al., 2021; HAN; KWAK; KIM, 2023), which is further described in Chapters 5 and 6.

2.2.2.1 generalized-Precision Time Protocol

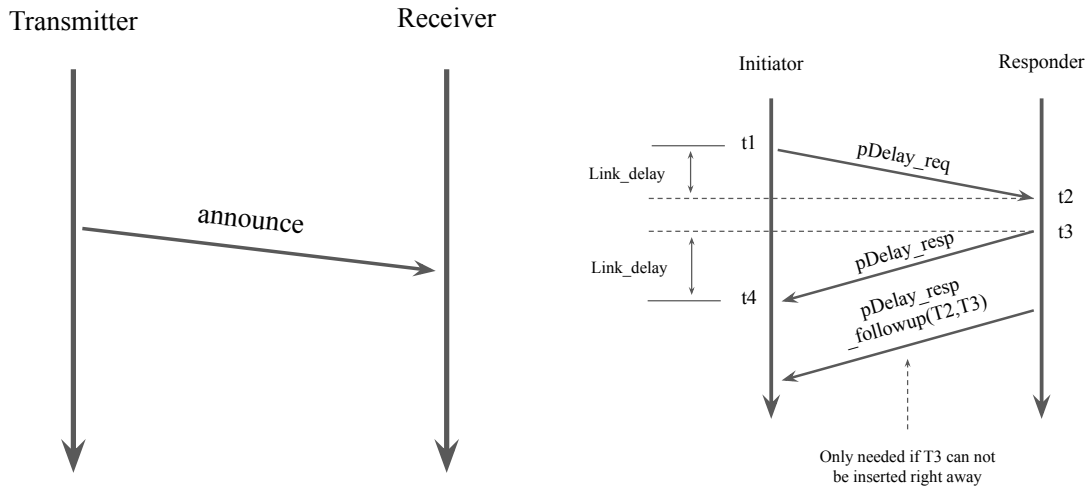
The main purpose of IEEE 802.1AS-2011/2020 or gPTP is to synchronize the nodes to a common reference time in an AVB cloud (a set of nodes that can communicate between themselves using the AVB/TSN protocols). This is considered one of the foundational protocols for automotive Ethernet, as one of the most important aspects of reliable media delivery is the synchronization of clocks across the network. To elucidate on the demanded precision, the gPTP standard mandates a precision of $\pm$ 500 μs for two end nodes that have fewer than 7 AVB nodes in between, which means that direct neighbors have to synchronize with nanosecond precision (MATHEUS; KÖNIGSEDER, 2021).

In gPTP, there are two main types of nodes: end stations, which are regular devices that run applications, and bridges, which are the switches that interconnect the end stations. Additionally, some nodes have the role of grandmaster, and they provide the reference clock to which the regular nodes will synchronize. The grandmaster can be pre- or auto-selected using the Best Master Clock Selection Algorithm (BMCA). In this algorithm, every node has the chance to become a grandmaster. The process starts with an "announce" message (Figure 8a), where the receiving node compares the information of the current grandmaster with its clock-related quality values. If eight differently read values of its clock yielded a better result than that of the current grandmaster, the node announces itself as the new grandmaster. "Announce" messages are sent cyclically, and the grandmaster can change during the network runtime (IEEE, 2020).

To achieve synchronization, gPTP measures the delay between the nodes, which is then used to calculate the synchronized time in the receiving systems. This synchronization is achieved with a technique called "two-step clock", which is performed as depicted in Figure 8b, where the initiator node sends a "pDelay_req" message, which is then followed by a "pDelay_resp" from the receiver. In cases where the Link_delay is fixed and symmetric, it can be computed by the equation (2.1). Clocks can be kept in sync by using the synchronized time from the grandmaster and adjusting for the delay between the neighbor nodes. The delay measurements occur periodically, such as the "announce" messages.

$$Link\_delay = \frac{(t_4 - t_1) - (t_3 - t_2)}{2}. \tag{2.1}$$

Figure 8 – Flow diagrams for gPTP protocol interactions.



(a) Announce message flow diagram. (b) pDelay measurement flow diagram.

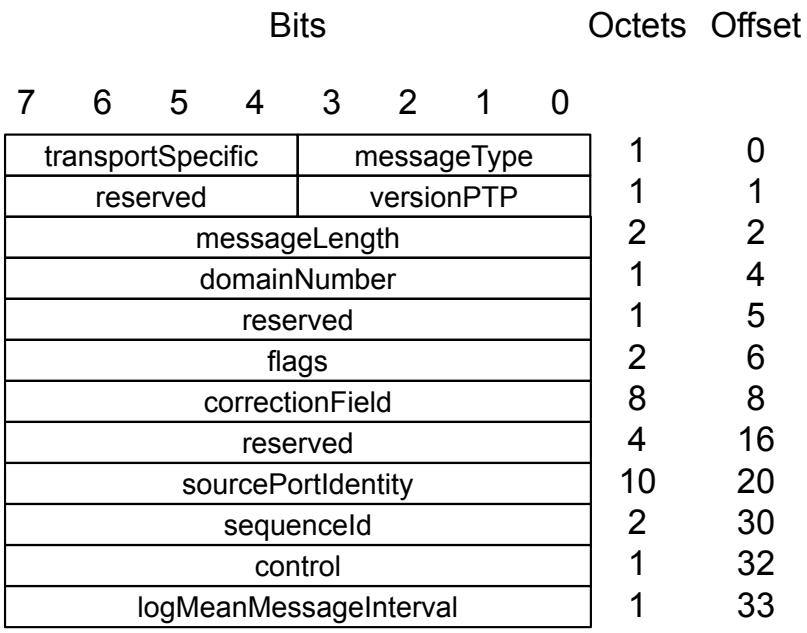
Source: Adapted from: (CORREA et al., 2014)

For automotive use, it is understandable to preselect the grandmaster and determine an ECU that every car is equipped with to be the grandmaster. Dynamic selection of the grandmaster has some drawbacks, such as slowing the start-up time and demanding more effort in the qualification and testing of the network. On the other hand, a pre-selection of the grandmaster would make it vulnerable to inconsistencies or issues with the grandmaster, where the network will not be able to elect a new grandmaster and generate error scenarios (CORREA et al., 2014).

On an Ethernet network, gPTP bridges and endpoints communicate via specially crafted frames, identified by the Ethertype 0x88F7. The gPTP messages are mainly made up of three sections: a header, which starts at the beginning of the payload field of the Ethernet frame and is common to all messages; a body, and then zero or more type length value (TLV) sections. The message header (presented in Fig. 9) contains several specific fields important for the synchronization message, such as the message type, that determines if it is a sync, pDelay_req, pDelay_resp, or other messages, and the flags and control, which are message dependent.

Security aspects. As previously discussed, one of the most important roles in gPTP is the grandmaster, as it determines the reference clock used by network common nodes. Therefore,

Figure 9 – gPTP message header.



Source: Adapted from: (CORREA et al., 2014)

if a malicious node has access to the network and can carry out a rogue master attack, where the malicious node might be elected as the new grandmaster, it can compromise the network clock reference (FOTOUHI et al., 2023). Another example is when a compromised node floods the network with synchronization messages, making it difficult for legitimate nodes to communicate between themselves. This specific scenario is later covered in Chapter 6 and is one of the attack scenarios described in (HAN; KWAK; KIM, 2023).

2.2.2.2 Audio/Video Transport Protocol

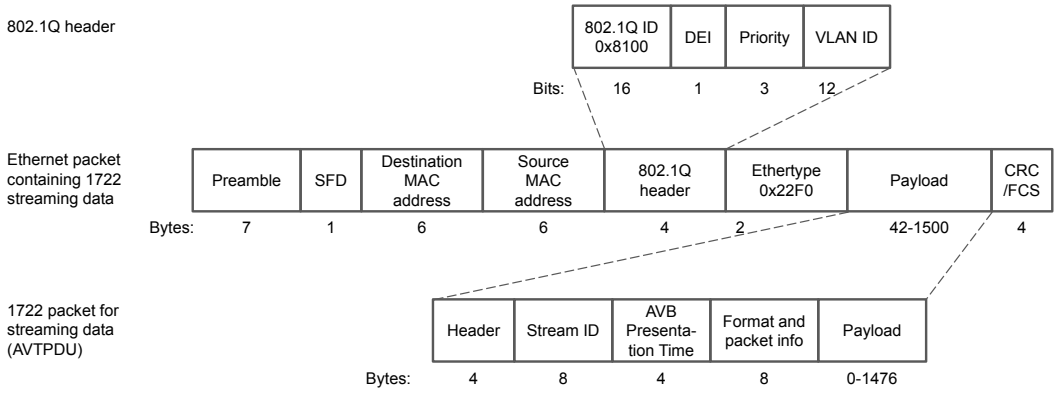
The core AVB protocols, such as SRP and gPTP, ensure the reliable transport of time-sensitive streams. However, they do not consider the nature of the payload being transported. In light of this, (IEEE, 2011) defines AVTP, which specifies how to transport AV data in time-sensitive networks. One of the key properties of the AVTP is that it identifies Ethernet packets carrying AV data on layer two, allowing it to bypass higher layer protocols, thereby reducing the processing time and making latency more predictable.

In Figure 10, we present the structure of AVTP packets. The packet is characterized by the 802.1Q ID of 0x8100 and the Ethertype of 0x22F0, which determines that AVTP is

being used. The Audio Video Transport Protocol Data Unit (AVTPDU) is within the Ethernet frame payload and contains the transmitted media data. There are two types of headers for AVTPDUs: the control header, which is used to create frames that handle the setup of a stream, and the stream header, which contains actual media. The IEEE 1722 packet comprises the following fields:

- Header: defines what type of AV data to expect. It also includes the sequence number to allow listeners to identify missing packets.
- Stream ID: defines a specific data stream and is derived from the talker's MAC address;
- AVB Presentation Time: defines when a received packet must be presented in the listener applications. This information reiterates the necessity of an accurately synchronized network to guarantee the data will be streamed in sync and also be used for feedback and correction for the synchronization;
- Format and packet info: defines the format and specific details of the data in the upcoming payload;
- Payload: contains the data itself.

Figure 10 – Audio/Video Transport Protocol frame.



Source: Adapted from: (MATHEUS; KÖNIGSEDER, 2021)

Initially, IEEE 1722-2011 (IEEE, 2011) mainly covered ISO 61883 headers. However, these headers did not comprehend formats such as MJPEG and H.264, which were discussed in the automotive industry for camera use cases. The scenario has changed with the IEEE 1722-2016 (IEEE, 2016) release, which introduced the previously mentioned formats and also included

the support for CAN, LIN, and FlexRay messages. Moreover, IEEE 1722 allows dynamic and static addresses, either multicast or unicast. Once the automotive scenario is not dynamic, a static pre-configuration is preferred to the IEEE 1722 address allocation for the nodes, as it also favors a short start-up time.

Security aspects. The AVTP protocol can carry sensitive information, such as video streams, which are crucial for autonomous vehicles and ADAS. In cases where a malicious node has access to the network, it can send manipulated or reinject previously transmitted data and eventually misguide the end node that will use this information for the vehicle decision-making process. In (JEONG et al., 2021), the authors presented a scenario of a replay attack in the AVTP protocol, where the malicious node sent outdated data that could delude the vehicle perception system and eventually lead to a crash.

3 IN-VEHICLE NETWORKS CYBERSECURITY

This chapter provides a brief overview of the cybersecurity scenario for IVNs, depicting the research and real-world cases involving automotive hacking. Additionally, we discuss existing protocols that cover authentication and encryption, followed by a detailed description of IDSs, focusing on their importance for enhancing security in IVNs and presenting a taxonomy that considers the target network layer, method, and design approach.

In the seminal work of (KOSCHER et al., 2010), the authors demonstrated that an attacker with access to the IVN could control a wide range of automotive functions, such as disabling the brakes and stopping the engine. Moving on, in (CHECKOWAY et al., 2011), the authors demonstrated the feasibility of remote exploitation of vehicles via connectivity tools, such as Bluetooth and cellular radio. Meanwhile, the authors of (MILLER; VALASEK, 2015) remotely hacked a Jeep Cherokee and turned off its engine on a highway, resulting in a recall of over 1.4 million vehicles. In the following years, other incidents and studies were reported and presented regarding vehicle vulnerabilities. In the subsequent years, more vulnerabilities were discovered in Tesla Model S (NIE; LIU; DU, 2017), Tesla Model X (NIE et al., 2018), and a BMW (CAI et al., 2019). More recently, (TINDELL, 2023) described how a Toyota RAV4 was stolen using a CAN injection technique, where the thieves accessed the physical CAN bus, impersonated the smart key ECU, and successfully opened and drove the vehicle. These studies and reports highlight the necessity for improved security mechanisms for IVNs.

Additionally, regulations were created to address cybersecurity when conceiving new vehicles and processes. For instance, ISO/SAE 21434 (STANDARDIZATION, 2021) focuses on cybersecurity in developing electrical and electronic systems for road vehicles. This standard offers guidance to ensure a shared understanding across the supply chain, enabling organizations to establish cybersecurity policies, mitigate cybersecurity risks, and promote a culture of cybersecurity awareness. Moreover, the (UN Regulation 155, 2021) specifies that vehicles manufactured after July 2022 in countries within the UNECE jurisdiction must be able to detect and report cyberattacks, among other requirements.

Furthermore, the MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) framework ((MITRE ATT&CK, 2024)) is a knowledge base of adversary tactics and techniques based on real-world observations. Although ATT&CK started for enterprise networks, it already has specialized versions for aerospatial ((SPARTA, 2024)) and automotive ((CyCraft Athena,

2024)) domains. For instance, in (VicOne, 2024), one of the Tesla hacks is dissected using an ATT&CK matrix focused on the automotive domain, depicting that techniques such as privilege escalation and code injection were used to affect a vehicle function and send manipulated CAN messages. Furthermore, the knowledge in ATT&CK matrices can contribute to the development of detection and analytics systems (WUNDER, 2020) and adversary emulation (STROM, 2020).

Moreover, vehicles are safety-critical devices with specificities that must be considered when designing and developing security mechanisms (VINCENZI et al., 2024). For instance, cars have a fixed topology of limited size and resources (such as memory and compute power); each model is designed once and built often, with a long product life cycle. If a vulnerability is found in a specific model, all cars of the same model become potential targets (MATHEUS; KÖNIGSEDER, 2021). For these reasons, we briefly summarize constraints of the automotive environment that must be addressed to design security mechanisms for IVNs (WU et al., 2020).

Hardware constraints. Despite the advances and upgrades in vehicles that enable applications like ADAS and even autonomous driving that demand hardware accelerators such as GPUs, most of the vehicle's ECUs are based on microcontroller devices, which are limited by memory and computation resources.

Cost constraints. A vehicle is a complex electronic system, and adding new components increases the cost of mass-produced automobiles. Therefore, including new hardware devices may not be desirable due to the increased total cost of the vehicle.

Detection accuracy and latency. Vehicles are safety-critical systems and cyberattacks in IVNs can lead to safety issues and harm people's lives. Therefore, it is important to accurately detect cyberattack incidents to prevent further issues while minimizing false positives. Additionally, the chosen security mechanisms should not introduce additional latency that could affect ECUs communication and the vehicle's functions.

3.1 SECURITY MECHANISMS

Software bugs, configuration errors, weak network design, and other issues can lead to vulnerabilities. Therefore, relying on a single security solution to handle all possible malicious activities is impractical. To combat this, the industry focuses on cybersecurity using multiple layers. This approach means that even if an attacker bypasses one layer, they will still have

to contend with several others. (MATHEUS; KÖNIGSEDER, 2021). In the upcoming paragraphs, we focus on the most used security protocols for the automotive environment.

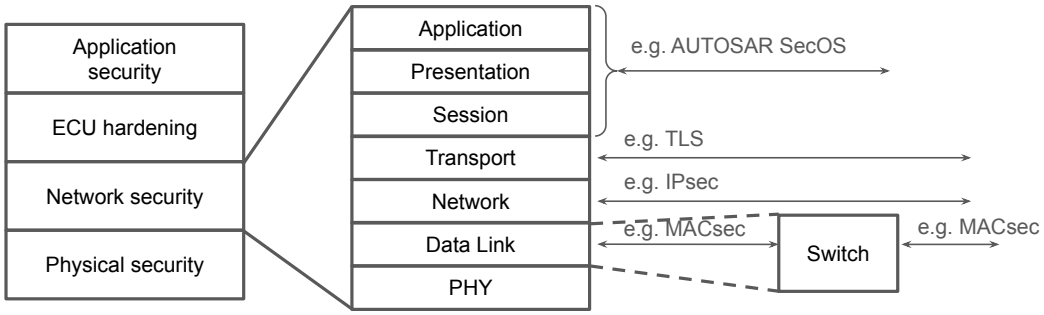
Figure 11 presents security mechanisms primarily based on authentication and cryptography for the automotive environment. The first one is the AUTOSAR Security Operations (SecOC), the first standardized security protocol for classical CAN bus networks, which can also be used on automotive Ethernet. SecOC ensures message authenticity, integrity, and freshness. AUTOSAR SecOC uses a freshness value (i.e., a timestamp or a counter) calculated and transmitted in each message. It also uses a message authentication code inserted into the message, which can be truncated or sent in separate messages in limited payload scenarios (LAUSER et al., 2024).

Media Access Control Security (MACsec) is a data link layer security protocol that provides message authenticity and data confidentiality specific to Ethernet frames. It uses the concept of secure channels to protect the links, where each channel has security associations that define the cryptographic keys used within the secure channel for authentication (IEEE, 2018). Furthermore, IPsec is a protocol suite used to secure communications by authenticating and encrypting each IP packet in a communication session. It works through a combination of protocols, where the authentication header (AH) ensures data integrity and authenticity by adding a header to the packet, but it does not encrypt the data. On the other hand, the Encapsulating Security Payload (ESP) provides confidentiality and authenticity by encrypting the data and adding a header for authentication. Before communication happens, a security association specifies how the data will be secured; this link is established through the Internet Key Exchange Protocol (IKE), which manages the keys to encrypt and decrypt the data (LAUSER et al., 2024).

Finally, the Transport Layer Security (TLS) protocol provides confidentiality, integrity, and authenticity. Its authentication is based on shared certificates during the handshake protocol, and its communication is secured in the record protocol. Despite being usually employed to secure external communication, it can also be used in internal applications such as ECU communication (RESCORLA, 2018).

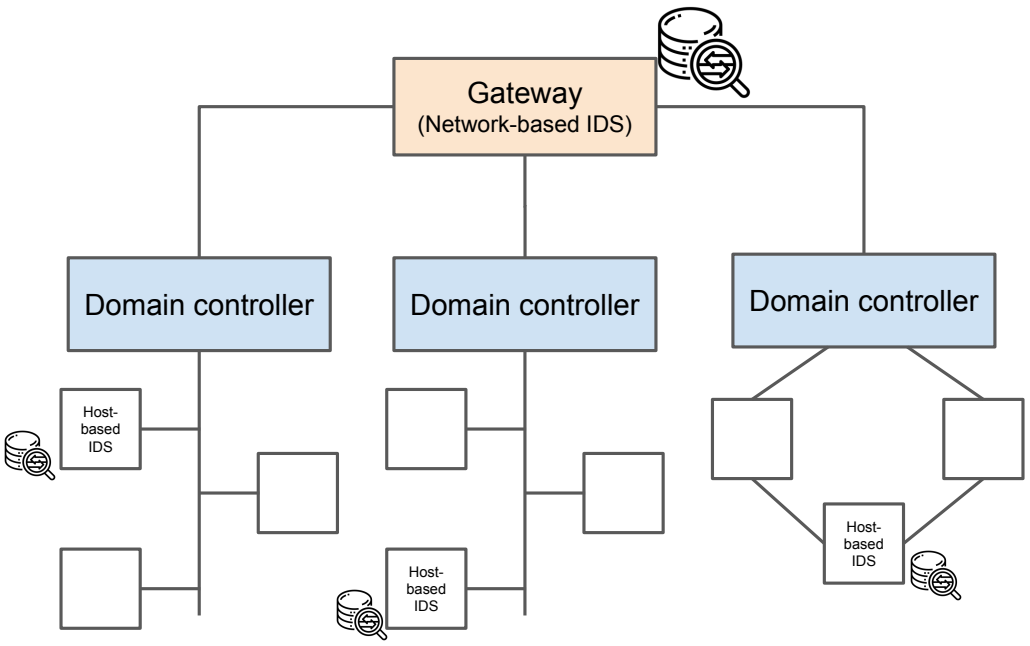
Furthermore, IDSs serve as a final line of defense in security, activated when all other security measures have failed. IDSs distinguish themselves from other security measures due to their cost-effective development and deployment compared to encryption and authentication solutions. This is because they do not require changes to message formats, add communication overhead, and can be deployed in a dedicated node. (WU et al., 2020). In Figure 12, we present

Figure 11 – Layered automotive cybersecurity approach and related mechanisms.



Source: Adapted from: (MATHEUS; KÖNIGSEDER, 2021)

Figure 12 – Deploy strategies for IDSs.



Source: The author (2024)

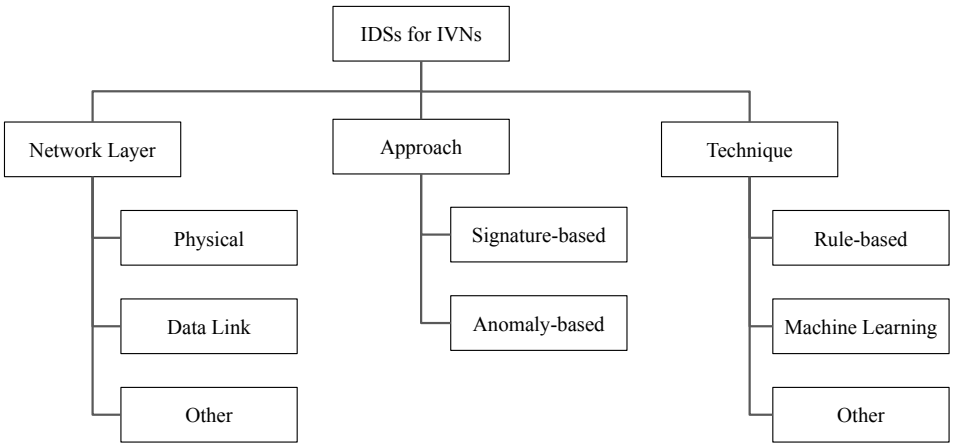
two approaches for automotive IDS deployment: in a central node, such as a network gateway, where it can monitor the entire network traffic (i.e., a network-based IDS), or in a specific end node, such as an ECU (i.e., a host-based IDS)(QUADAR et al., 2024).

This work focuses on proposing and evaluating intrusion detection systems for automotive Ethernet networks that consider automotive environment constraints, such as low detection time and high detection accuracy (WU et al., 2020). In the next section, we present a taxonomy for IDSs focused on IVNs based on their network protocol layer, approaches, and techniques.

3.2 INTRUSION DETECTION SYSTEMS FOR IVNS TAXONOMY

Figure 13 presents our IVN IDS taxonomy, specifying the network protocol layers used as the data source, the IDS approach for representing attacks and normal packets and the employed technique to design the IDS. These classifications are discussed in the following subsections.

Figure 13 – Taxonomy for IVN IDSs based on the network layer, approach, and technique.



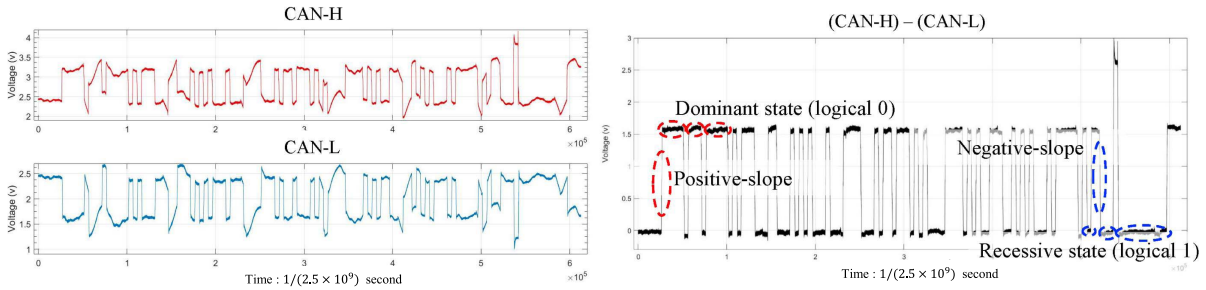
Source: The author (2024)

3.2.1 Network Layer

The first item in the taxonomy considers the network layer to which the IDS is attached. The discussion focuses on the two main layers used in state-of-the-art work: the physical and data link layers.

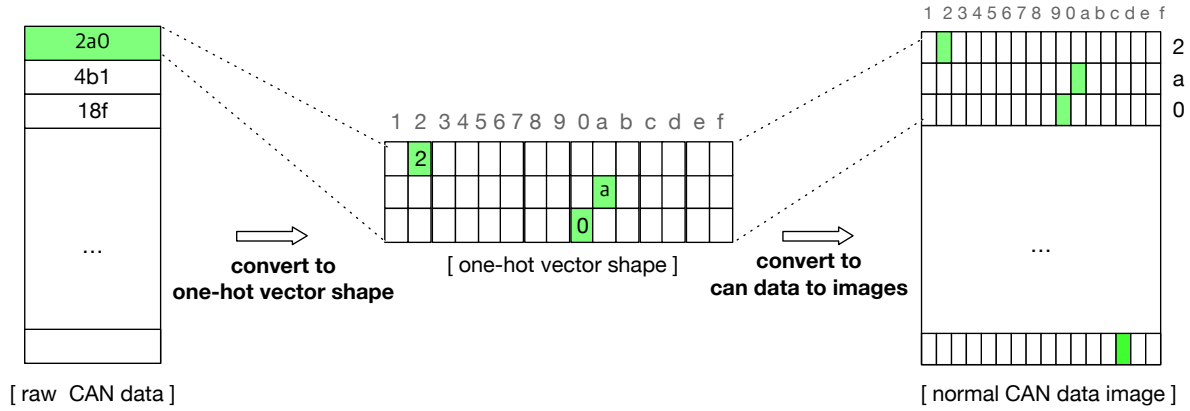
Physical Layer. IVN IDS attached to the physical layer use patterns identified at the bus level, such as clock deviations (CHO; SHIN, 2016) and the bus's fingerprint characteristics based on voltage measurements (CHO; SHIN, 2017). In (CHOI et al., 2018), the authors proposed an IDS that uses features based on statistics obtained from the CAN bus electrical signals characteristics (presented in Figure 14) such as the dominant and recessive states and positive and negative slopes to identify cyberattacks. Despite the high accuracy characterization, these IDSs usually require specific hardware to capture the low-level signal characteristics and are ineffective for detecting attacks at higher layers.

Figure 14 – Electrical CAN Signal.



Source: (CHOI et al., 2018)

Figure 15 – Process of converting CAN IDs to images.

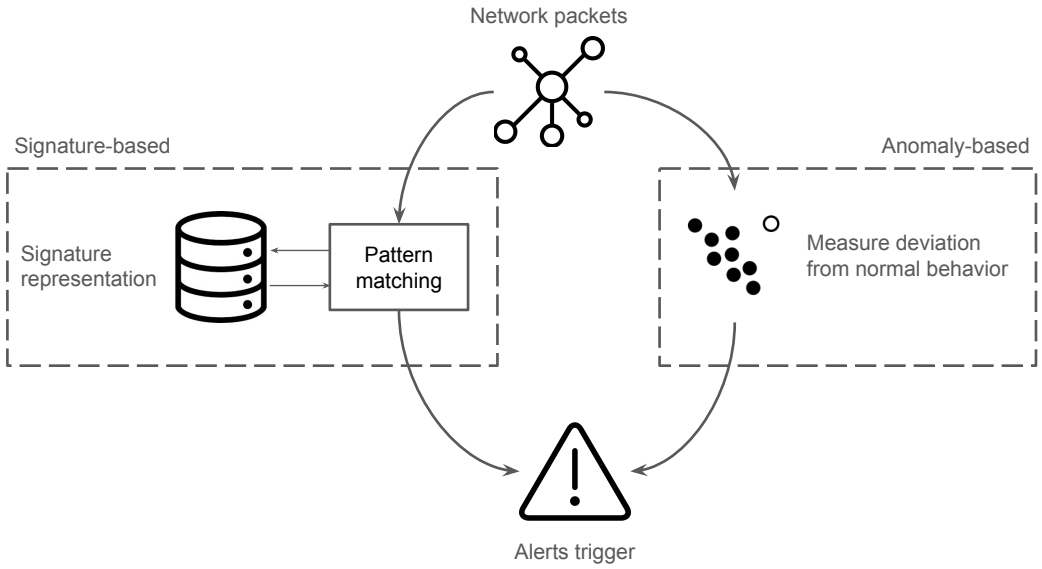


Source: (SEO; SONG; KIM, 2018)

Data Link Layer. Most of the IVN IDSs use data from the data link layer as their main information to detect cyberattacks. The information can be at the message level, such as time intervals and frequency, to the entire packet, including its IDs, source and destination addresses, and payloads. For example, in (JEONG et al., 2021), the authors used the raw automotive Ethernet packets bytes as input for their feature generator, transforming the data to serve their IDS. The same thing happened for the IDS proposed in (HAN; KWAK; KIM, 2023), where the authors used the raw network packets from multiple protocols to build features depicting the protocol transitions and the information in the packets. Additionally, for the CAN protocol, several works use the data from the CAN frame to identify cyberattacks, such as in (SEO; SONG; KIM, 2018), which uses the CAN IDs to create images, as depicted in Figure 15, which will further be used as input to their proposed a GAN-based IDS.

Other layers. Despite most works being mainly based on the initial layers protocols, some

Figure 16 – Different approaches to design an IDS.



Source: The author (2024)

works consider upper layers protocol. For instance, the IDS proposed in (ALKHATIB; GHAUCH; DANGER, 2021) uses data from the application layer protocol SOME/IP to detect different cyberattacks in an automotive Ethernet network.

3.2.2 Approach

This section covers the design approaches of IDSs, as presented in Figure 16. It defines how IDS distinguishes normal from malicious traffic, independent of the employed technique to achieve this goal.

Signature-based. Signature-based IDSs create a signature (or representation) of the events they are designed to detect: cyberattacks and regular packets. One of the advantages of signature-based IDSs is their ability to detect both cyberattacks and regular frames/packets effectively. However, this approach cannot detect zero-day attacks, a major threat due to the constant development of new cyberattacks. Additionally, the signature representation must be regularly updated, which is challenging as new attacks are constantly emerging, and data acquisition and labeling are costly.

Anomaly-based. Anomaly-based intrusion detection systems (IDSs) focus on creating a model of a network's typical behavior and then analyzing whether new packets deviate from this

normal behavior. These deviations often indicate an attack. This approach overcomes one of the limitations of Signature-based IDSs, as it does not rely on known cyberattack signatures, making it capable of detecting zero-day threats. However, this approach may result in a slight drop in detection metrics, particularly in cases where cyberattacks are more complex to detect and exhibit behavior similar to regular network traffic. Additionally, this method tends to have a higher rate of false positives and requires attack-free data for training.

3.2.3 Techniques

This section covers the typical techniques used in state-of-the-art work to implement the decision-making process of the IDSs. The techniques specify the algorithm used to implement a specific approach for developing the IDSs.

Rule-based. Rule-based IDSs are a traditional approach to developing IDSs. In this method, security experts create rules based on previous attack patterns and behaviors. The main advantages of this approach are that it is easy to understand the IDS's decision-making process, and efficient at detecting known attacks. However, there are several limitations to rule-based IDSs. These include the need for frequent rule updates and their inefficiency in detecting unknown or zero-day attacks. In (GMIDEN; GMIDEN; TRABELSI, 2016), the authors proposed a rule-based system that monitors the time difference between consecutive CAN frames. If the interval exceeds the established "normal" value, it increments an anomaly score. Then, when the anomaly score is higher than a pre-defined threshold, an alert is raised. Similar techniques are also proposed in (SONG; KIM; KIM, 2016) and (LEE; JEONG; KIM, 2017).

Machine Learning. ML and DL-based IDSs have gained attention for achieving great detection results in complex network scenarios such as IVNs for their ability to identify and model complex patterns in high-dimensional data (SEO; SONG; KIM, 2018; HAN; KWAK; KIM, 2023; JEONG et al., 2021). However, ML models rely on representative and labeled data for training and evaluation. Additionally, unlike rule-based techniques, most DL models are black-box, making it challenging for humans to interpret and comprehend how the model used the data to make its decision (JACOBS et al., 2022b).

Other techniques. Here, we have compiled various techniques that are also used to develop IVN IDSs that do not fit into the previous categories. For instance, in (LAMPE; MENG, 2022) and (MARCHETTI; STABILI, 2017), the authors use CAN bus historical data to create a repre-

sentation of "normal" ID transitions. In cases where the IDS sees a transition that is not in the normal representation, it considers the traffic anomalous. Moreover, in works such as (WANG; LU; QU, 2018) and (WU et al., 2018), the authors calculate the entropy of CAN IDs to establish a "normal" entropy; if the following entropy computations do not fall in the "normal" entropy range, the traffic is considered malicious.

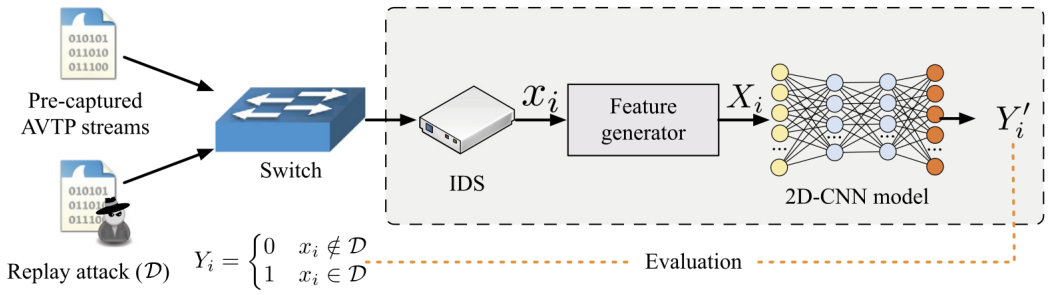
To summarize the characteristics for proposing novel IDSs for automotive networks, it is essential to consider the environmental constraints, such as hardware, cost, and detection accuracy. After considering these constraints, we should make design decisions regarding the network layer data, approach, and technique.

4 RELATED WORK: IDS FOR AUTOMOTIVE ETHERNET

This chapter describes the current state-of-the-art research that addresses intrusion detection in automotive Ethernet networks. We describe the methods proposed by the authors, the contributions of their work, and the limitations that open up possibilities for additional exploration.

The authors of (JEONG et al., 2021) investigated the issue of intrusion detection in automotive Ethernet networks. In their work, the authors proposed the IDS presented in Figure 17, which is based on a feature generator and a two-dimensional-convolutional neural network (2D-CNN). The IDS gathers AVTP packets to create a network traffic image used as input for the feature generator. Their proposed IDSs aims to detect replay attacks in AVTP packets. The conducted replay attack consists of a specific video frame repeatedly transmitted in the network. Although the obtained results have shown that the IDS can accurately detect most of the malicious frames, the IDS needs to be executed in Graphical Processing Units (GPUs) to achieve real-time detection (a detection before the next frame arrives), which leads to high deployment cost.

Figure 17 – CNN-based IDS for detecting replay attacks in AVTP.

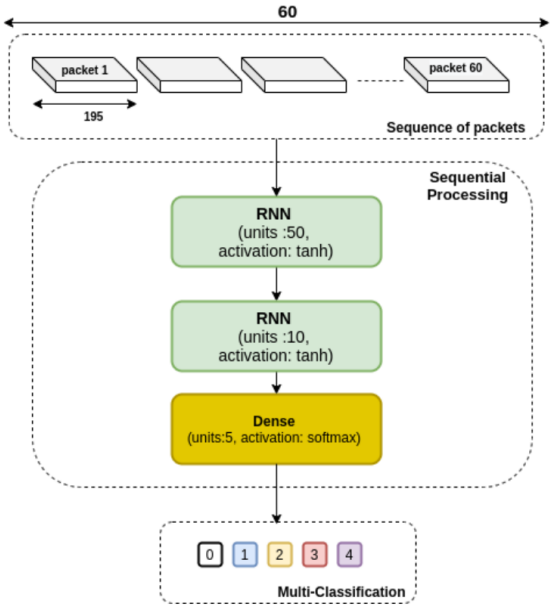


Source: (JEONG et al., 2021)

Moving on, the authors of (ALKHATIB; GHAUCH; DANGER, 2021) addressed the problem of attack classification in scalable service-oriented middleware over IP (SOME/IP) networks. Their proposed IDS is based on a recurrent neural network (RNN) to better represent the temporal correlation between the packets in a session of the SOME/IP protocol (as depicted in Figure 18). Their dataset was generated synthetically using Python libraries and contains normal packets and attack scenarios such as request without response and response without request. However, their work focused on evaluating the performance of RNNs for attack classification in an offline intrusion detection scenario, i.e., the attacks are only detected after

a SOME/IP session has ended. Another downside is that their method is made specifically SOME/IP networks and cannot be used with other protocols.

Figure 18 – RNN-based IDS for classifying attacks in SOME/IP protocol.



Source: (ALKHATIB; GHAUCH; DANGER, 2021)

In (CARMO et al., 2022), the authors proposed the use of XGBoost algorithm to identify replay attacks in AVTP packets. They have evaluated their proposed IDS using the automotive Ethernet intrusion dataset (AEID) developed in (JEONG et al., 2021). Their proposal detected cyberattacks quickly (620 µs/sample) and accurately, using a low-cost, CPU-based hardware such as a Raspberry Pi 3. However, the detection results obtained by the authors when using a model that is not adequate to detect spatial or time relationships in data highlight that the replay attack scenario proposed in (JEONG et al., 2021) of injecting the same video packet every time may be easily detectable when considering supervised training.

The authors of (ALKHATIB et al., 2022) were the first to consider the approach of anomaly-based IDSs for automotive Ethernet networks. The authors have evaluated different machine learning and deep learning algorithms for attack detection using the AEID dataset. Their proposed IDSs are based on autoencoder(AE) models such as the convolutional autoencoder (CAE) and a long short-term memory-based autoencoder (LSTM-AE), which achieved better detection time and detection accuracy metrics when compared to other traditional anomaly detection models such as one-class support vector machine (OCSVM) and isolation forest. AE architectures are DL-based anomaly detection methods that use an encoder to create a latent space representation of the data and a decoder to reconstruct the data based on

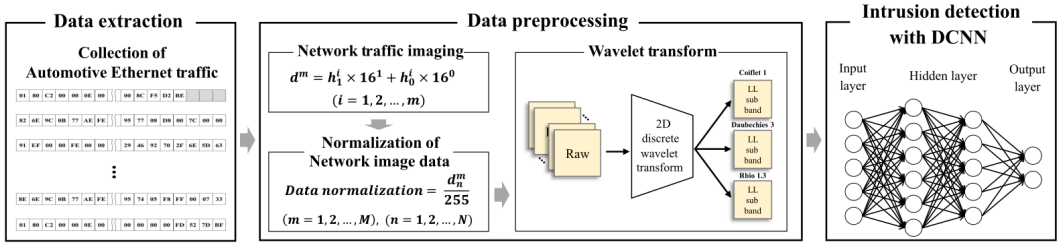
this representation. If the reconstructed data differs significantly from the original data, those samples can be identified as anomalous, potentially indicating an attack. Despite the obtained results demonstrating the suitability of AE models for detecting cyberattacks, the LSTM-AE and CAE models had a slight decrease in detection performance when compared to other state-of-the-art works (JEONG et al., 2021; CARMO et al., 2022). The authors highlighted the need for an automotive Ethernet dataset with diverse types of attacks to extrapolate the IDS evaluation to attacks other than solely replay attacks.

Furthermore, the authors of (HAN; KWAK; KIM, 2023) proposed the IDS architecture depicted in Figure 19a, which uses three different wavelet transform filters concatenated with a CNN architecture comprising point and depth-wise convolutions and short connections to detect cyberattacks in the heterogeneous automotive Ethernet network (presented in Figure 19b). The authors evaluated their IDS using the TOW-IDS dataset, which contains packets from AVTP, CAN over user datagram protocol (UDP), and gPTP. Their new dataset comprised several new cyberattacks, such as CAM table overflow, PTP synchronization, AVTP frame injection, and CAN DoS and replay attacks, overcoming the one attack limitation of the AEID dataset. Although the authors created a novel dataset with several attack scenarios, their labeling criteria and IDS only considered frames malicious or not, without the ability to classify the kind of attack. Alongside, the authors only compared their results with traditional CNN architectures and not with other automotive Ethernet state-of-the-art IDSs.

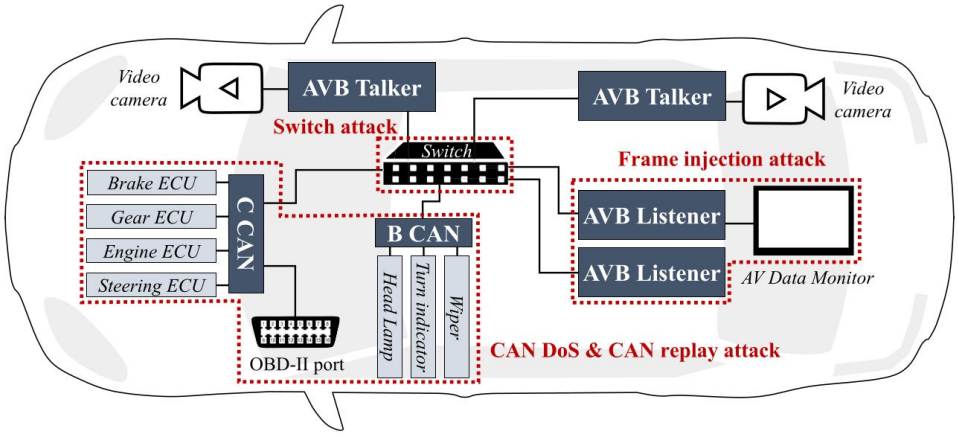
The authors of (SHIBLY et al., 2023) proposed a technique that demands less labeled data to train their proposed IDS in a semi-supervised manner. The authors leveraged the use of Generative Adversarial Networks (GAN) to perform data augmentation and enable their IDS to be trained using only 10% of labeled data. Their proposed IDS was evaluated with the TOW-IDS dataset and several other CAN datasets to assess the feasibility of the semi-supervised training. Despite the reduced amount of labeled data needed to develop the IDS, the obtained results still need further improvement to match the detection performance of other methods, such as the one proposed in (HAN; KWAK; KIM, 2023).

At last, the authors of (JEONG et al., 2023) proposed the multimodal feature extractor and an unsupervised IDS depicted in Figure 20. The multimodal feature extractor gathers information from a protocol transition matrix, raw packet payloads, and packet timestamps statistics. The authors have proposed using an encoder and decoder during the training phase to enable unsupervised training. The decoder is later replaced with a mapper that aggregates the data generated from the encoder and determines a threshold to distinguish normal and

Figure 19 – Proposed IDS architecture containing wavelet transform, CNN, and the heterogeneous automotive Ethernet network used to develop the TOW-IDS dataset.



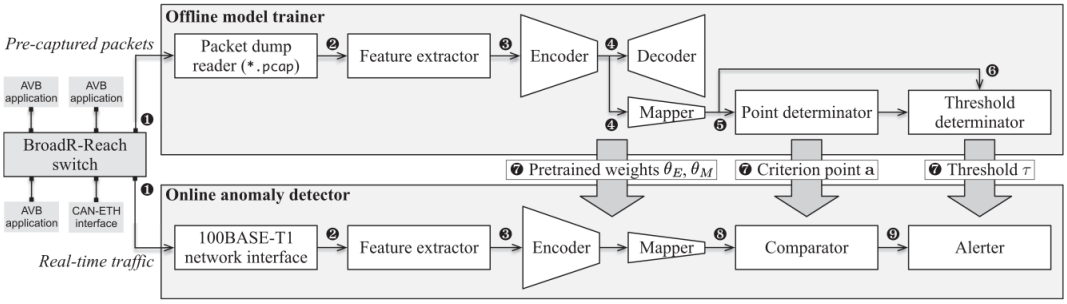
(a) IDS based on three wavelet transform and CNN.



(b) Heterogenous automotive Ethernet network containing different cyberattacks and protocols.

Source: (HAN; KWAK; KIM, 2023)

Figure 20 – Architecture of the IDS with a multimodal feature extractor, an encoder, and a point mapper to distinguish normal from anomalous traffic.



Source: (JEONG et al., 2023)

anomalous traffic. The authors evaluated their proposed IDS with the TOW-IDS dataset and analyzed the detection rates per attack basis. Despite the ability to detect novel attacks, their proposed IDS cannot classify the kind of attack and demands to be deployed in GPU devices to achieve real-time detection.

Table 3 summarizes the works mentioned above, highlighting their methods, datasets, and

key characteristics. The timing requirements in the table are regarding the real-time detection threshold mentioned in (JEONG et al., 2021) of 1,735 μ/sample during a replay attack. It also includes the description of the IDSs proposed in this dissertation and where they stand regarding the considered characteristics.

Table 3 – Related work. AEID and TOW-IDS refer to the datasets proposed in (JEONG et al., 2021) and (HAN; KWAK; KIM, 2023), respectively. SOME/IP refers to the dataset used in (ALKHATIB; GHAUCH; DANGER, 2021)

Reference	Method	Dataset	Supervised	Multi-label	Timing requirements
(JEONG et al., 2021)	2D-CNN	AEID	Yes	No	Yes, with GPU devices
(ALKHATIB; GHAUCH; DANGER, 2021)	DL methods	SOME/IP	Yes	No	No, offline IDS
(CARMO et al., 2022)	XGBoost	AEID	Yes	No	Yes
(ALKHATIB et al., 2022)	CAE and LSTMAE	AEID	No	No	No
(HAN; KWAK; KIM, 2023)	Wavelet transform feature extractor and customized DCNN	TOW-IDS	Yes	No	Yes, but it is not clear in which device
(SHIBLY et al., 2023)	Feature-aware semi-supervised learning	TOW-IDS	Partially	No	Not mentioned
(JEONG et al., 2023)	Multimodal feature extractor with a neural network	TOW-IDS	No	No	Yes, with GPU devices
Chapter 5 proposed system	**Pruned and quantized 2DCNN**	**AEID**	**Yes**	**No**	**Yes**
Chapter 6 proposed system	**Multi-stage IDS**	**AEID and TOW-IDS**	**Yes**	**Yes**	**Yes**

Source: The author (2024)

Targeting the issue of a balancing detection time and overall detection metrics, the Chapter 5 of this dissertation covers the proposal of using a technique that optimizes detection accuracy, detection time, and model size during the IDS training phase. The main motivation is to achieve an IDS with a low detection time and storage size, enabling it to be deployed in resource-constrained devices. Although we obtained a reduction of 900x in the model size compared to the work of (JEONG et al., 2021), there is still room for improvement in the detection time.

In addition, Chapter 6 describes the proposal of a system that stands out from the existing works in detecting cyberattacks quickly and efficiently. It uses a divide-and-conquer strategy based on using a traditional ML algorithm and a more robust DL algorithm concurrently to identify suspicious events and then accurately classifies them among the known cyberattacks. The attack classification information can be communicated to the user and other systems in real-time to prevent further damage and improve forensics.

5 MULTI-CRITERIA OPTIMIZED DEEP LEARNING-BASED INTRUSION DETECTION SYSTEM FOR DETECTING CYBERATTACKS IN AUTOMOTIVE ETHERNET NETWORKS

This chapter contains sections three to five of the paper "Multi-Criteria Optimized Deep Learning-based Intrusion Detection System for Detecting Cyberattacks in Automotive Ethernet Networks"(LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023) accepted and presented in Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) 2023.

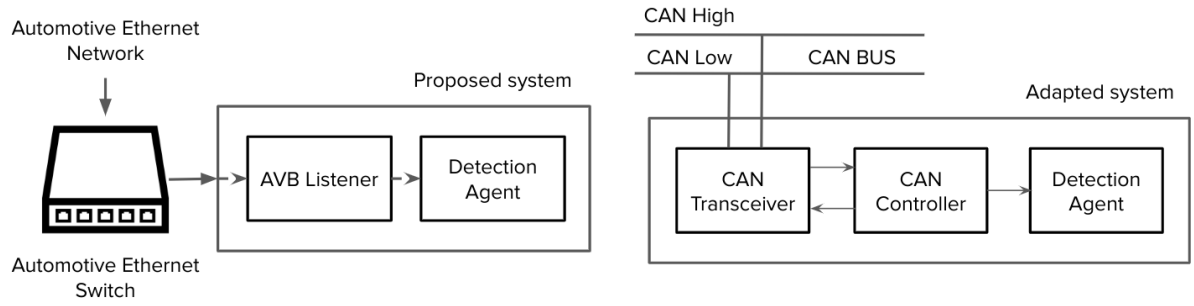
5.1 PROPOSED SYSTEM

In this section, we propose a DL-based IDS whose primary goal is to detect replay cyberattacks in an AVTP network, that is, classify a packet as benign or malign. Besides the correct packet classification, our secondary goal is to generate an optimized model regarding its detection time and storage size, to have a system that enables real-time detection and is more suitable to be deployed in in-vehicle network environments.

Our IDS uses the 2D-CNN and feature generator proposed in (JEONG et al., 2021) as its reference architecture and pre-processing step, respectively. Alongside, our IDS uses the multi-criteria optimization technique proposed in (GIRISH et al., 2022) (further described in Sub-section 5.1.3) that improves the model detection metrics, detection time, and storage size simultaneously during the training step. Once the training step is concluded, an optimized model is obtained, which has fewer internal connections and uses fewer bits to represent its weights when compared with the reference architecture. The optimized model will be further used as the deep learning algorithm of the detection agent of our IDS.

Our architecture comprises an AVB listener and a detection agent (presented in Figure 21). The AVB listener acts as a media converter to read the AVB packets sent by the automotive Ethernet switch. Finally, the detection agent deploys the optimized deep-learning algorithm that detects cyber-attacks. The detection is triggered when a total of “window_size + 1” packets are grouped and provided to the feature generator. When a frame is considered malicious, the IDS informs that the network is under attack. The algorithm of our proposed IDS is presented in Algorithm 1. In our scenario, the “window_size” value was 44. This value was chosen via hyperparameter tuning in (JEONG et al., 2021), focusing on balancing accuracy and F1-score in the test set, as well as a fast training time.

Figure 21 – On the left is our proposed IDS architecture to be deployed in an automotive Ethernet environment. On the right is how our architecture could be adapted to be deployed in a CAN bus environment.



Source: The author (2024)

Algorithm 1 Proposed Multi-criteria Optimized IDS

```
w: Window size = 44
Train the detection agent's deep learning model
Obtain the optimized deep learning model from the training step
while The automotive Ethernet switch receives AVTP packets do
    group a set of w + 1 sequential AVTP packets in an array X
    uses the array X as input of the Feature Generator (Sub-section 5.1.2) and get feature
    use feature as input to the optimized deep learning model
    if The frame is considered malicious then
        The detection agent sends a signal informing that the network is under attack
    end if
end while
```

It is worth mentioning that our detection agent (feature generator and deep-learning model) could be further used in other in-vehicle networks, such as CAN. To illustrate the possibility of use in other networks, we present an adapted architecture on the right side of Figure 21 of how our detection agent could be deployed in a CAN network. It is essential to mention that it would be necessary to fine-tune the feature generator and retrain the deep-learning method for this new scenario.

5.1.1 Attack model

To execute a replay attack, an attacker initially must have access to the network (in our case, an IVN) and sniff a group of packets or already possess pre-captured packets. With the packets in hand, the attacker resends them to the network to confuse the nodes that use this information. In the case of connected vehicles, where the perception system is responsible for driving decision-making, a replay attack may endanger the life of the driver and the people

around them.

An example of how the replay attack could be harmful is presented in Figure 22. Here, the camera ECU captures the road traffic images, and the ADAS detects three people crossing the street, which triggers a slow down or stop command for the powertrain ECU. If a replay attack happened at this moment, the vehicle would be misguided by the received image that shows no one crossing the street and would continue driving, which could lead to an accident.

Figure 22 – On the left is the original frame, where the people crossing the street are detected by the ADAS. On the right is the frame received during a replay attack, where the vehicle is misguided to see no one crossing the street.

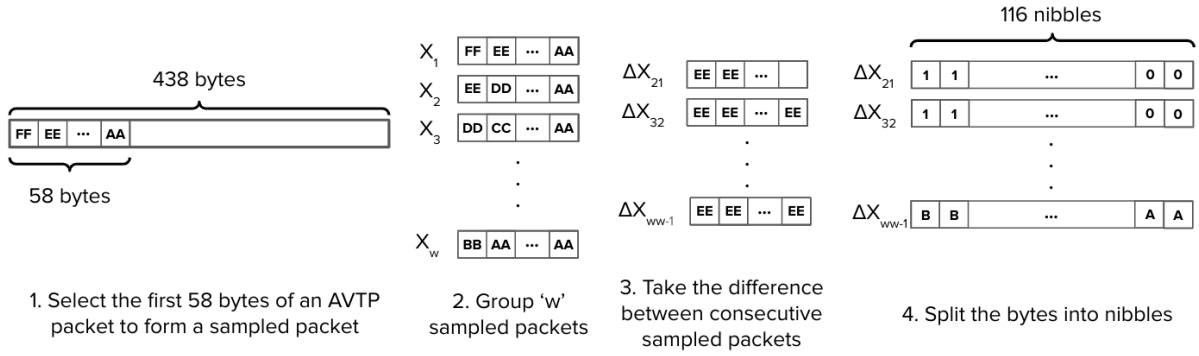
Source: Adapted from (BURKE, 2019)

5.1.2 Feature generator

For the feature generator, we have chosen to maintain the one initially proposed in (JEONG et al., 2021), which is briefly described in this section. An AVTP packet contains 438 bytes, but the authors of (JEONG et al., 2021) found that only the first 58 bytes contained interesting information regarding its header and payload fields. For the sake of simplicity, we refer to these first 58 bytes as “sampled packet".

Once the sampled packets are gathered, they are aggregated into groups of w sampled packets, where w is called the “window size". Once these groups are established, the byte-per-byte difference between each consecutive sampled packet is taken. At last, the bytes are split into nibbles, and each group is considered an input sample for training the IDS. This feature generator process is illustrated in Figure 23.

Figure 23 – The steps of the feature generator.



Source: The author (2024)

5.1.3 Optimization technique

The optimization technique considered in the proposed IDS is the LilNetX framework, initially proposed in (GIRISH et al., 2022). This technique optimizes the storage size, detection time, and detection metrics simultaneously during the training step, preventing the need for a post-training step such as quantization and pruning methods. LilNetX is based on a loss function that updates the network weights Θ according to:

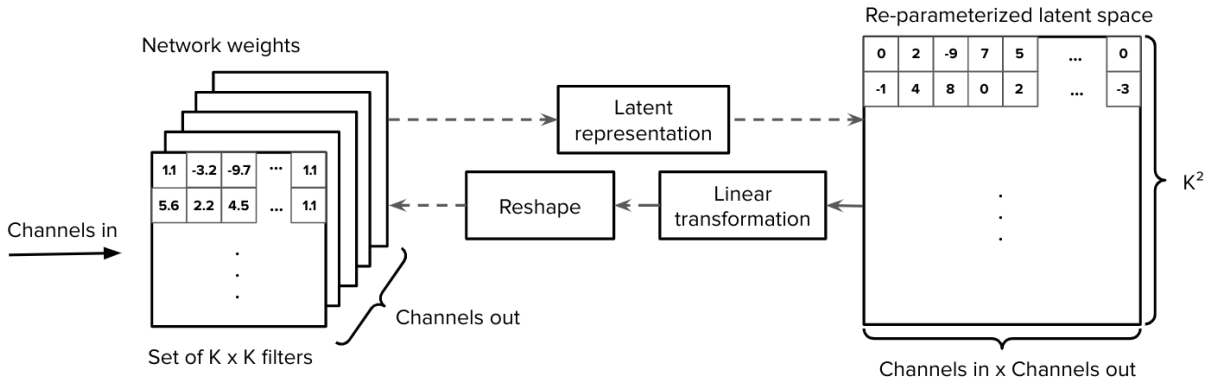
$$\mathcal{L}(\Theta) = \mathcal{L}(\Theta)_{detection} + \mathcal{L}(\Theta)_{storage\ size} + \mathcal{L}(\Theta)_{detection\ time}, \tag{5.1}$$

the detection term uses cross-entropy to improve the model's ability to classify the samples correctly. The storage size term relies on the latent space representation of the network weights, which is obtained by adapting the compression technique proposed in (OKTAY et al., 2019). Such representation has a learned probability model of the network weights and imposes a small bit representation based on an entropy penalty. The detection time term introduces sparsity also using entropy but bringing the latent space parameters to zero to reduce the number of computations. At last, the zero-valued parameters are brought back to the original network weights by using a linear transformation without linear coefficients. The transformation process of the network weights is illustrated in Figure 24.

5.1.4 Deep-learning model architecture

The reference architecture for our IDS was the 2D-CNN originally presented in (JEONG et al., 2021) and briefly described in Table 4. The main differences in our architecture are regarding

Figure 24 – The transformation process of network weights shape during the LilNetX framework optimization.



Source: The author (2024)

the use of the LilNetX framework, where we have added the boundary hyperparameter, which is responsible for controlling the variance of the network weights and plays a significant role in the convergence of the optimization technique. According to (GIRISH et al., 2022), the boundary must be greater than 0.5. The authors of (GIRISH et al., 2022) state the network's weights initialization follows a uniform probability distribution, and the weights must be greater than 0.5 to avoid being directly rounded to 0 during the optimization process. We have experimented with boundary values in the interval of 0.55 and 0.75. We have also added weight decoders to the convolutional and dense layers, where these decoders will map the layers' weights in the latent space representation using a probability model to achieve a small bit representation.

Table 4 – 2D-CNN architecture and hyperparameters.

Layer name	Activation	Regularization	Hyperparameters	Weight Decoder
Conv2D_1	ReLU	Batch Norm	in_ch=1, out_ch=32, kernel_size=5, stride=1, padding='same'	Conv5x5
MaxPool_1	-	-	kernel_size=2, stride=2	-
Conv2D_2	ReLU	Batch Norm	in_ch=32, out_ch=64, kernel_size=5, stride=1, padding='same'	Conv5x5
MaxPool_2	-	-	kernel_size=2, stride=2	-
Flatten	ReLU	Dropout	in_feat=20416, out_feat=64	Dense
Dense	Sigmoid	Dropout	in_feat=64, out_feat=1	Dense
Output	-	-	-	-

Source: The author (2024)

5.2 METHODOLOGY AND EXPERIMENTAL EVALUATION

This section provides the methodology and experimental setup used to evaluate our proposed IDS. We made our code available at https://github.com/luigiluz/multi-criteria-dl-based-

ids-for-automotive-ethernet to ease the reproduction of our experimental results.

We have chosen the Python programming language and the PyTorch framework due to their vast use for machine learning and deep learning applications and their wide documentation and community. The experiments were conducted in Google Colab Pro GPU NVIDIA Tesla P100 for the training and validation steps related to detection metrics. For the timing metrics, it was used an Intel(R) Xeon(R) CPU @ 2.20GHz.

5.2.1 Dataset presentation and preparation

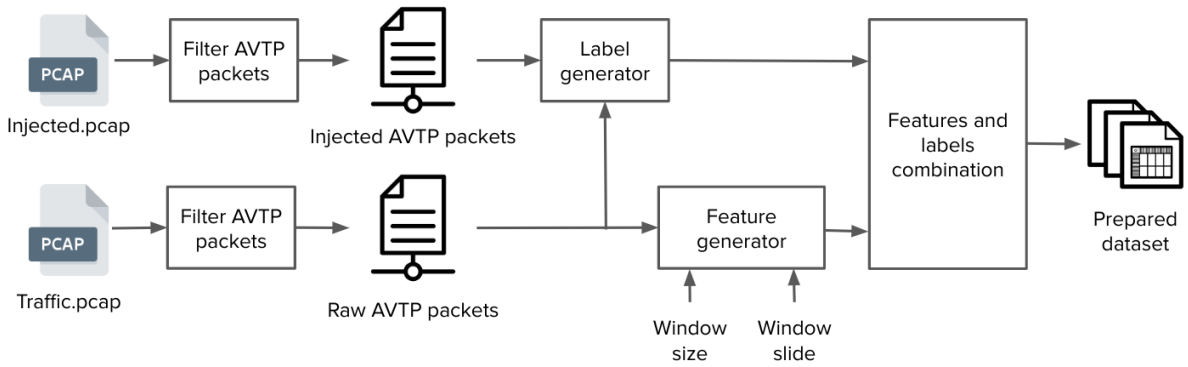
The dataset used to evaluate our proposed IDS is the publicly available AVTP intrusion dataset (JEONG et al., 2021). We have chosen this specific dataset to evaluate our results because it is the most used dataset regarding automotive Ethernet intrusion detection works (JEONG et al., 2021; ALKHATIB et al., 2022; CARMO et al., 2022), which enables the comparison of our results with more existing works.

It consists of real network traffic, available in .pcap files, which were collected from a vehicle containing a camera that captured video streams and sent them over as AVTP packet payloads. The authors collected data in two different scenarios: in a laboratory environment (referred to as $D_{indoors}$) and in a real-world traffic environment (referred to as $D_{outdoors}$). These packets were made available in different .pcap files that need to be combined and preprocessed.

As the datasets are made available in .pcap files, it is necessary to prepare them to be used to train and evaluate our IDS. This preparation process is presented in Figure 25. The first step is to filter only the AVTP packets. The filtering process is done by choosing the packets that have only 438 bytes in length. After that, the labels are generated to classify the packet as “benign"or “malign". If the corresponding raw AVTP packet is the same as one of the injected AVTP packets, it is considered “malign", otherwise it is “benign". In sequence, the corresponding features are generated (applying the method explained in Sub-section 5.1.2). At last, the features and labels are combined to generate the prepared dataset. This preparation process is executed once for indoor and outdoor data.

The resulting samples distribution of each prepared dataset is: $D_{indoors}$ has 446,372 benign packets and 196,892 malign packets, while $D_{outdoors}$ has 1,494,253 and 376,236 benign and malign packets, respectively.

Figure 25 – AVTP Intrusion dataset preparation process.



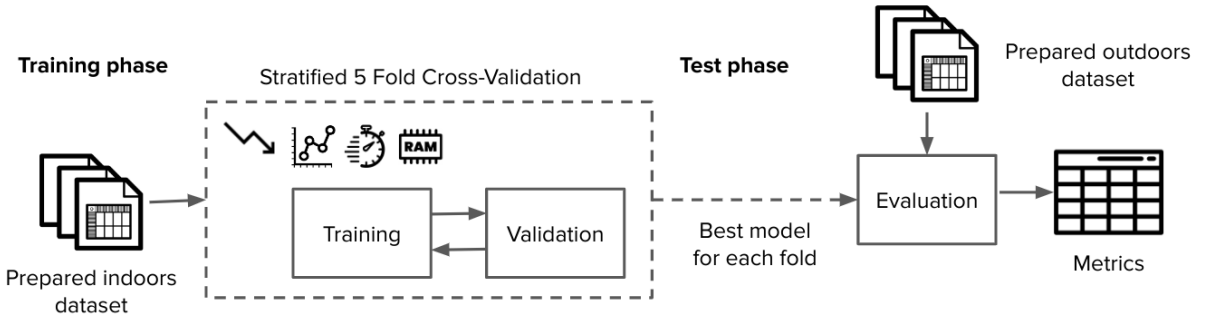
Source: The author (2024)

5.2.2 Experimental evaluation

The experimental evaluation of our IDS is divided into two phases: the training phase and the test phase. In the training phase, only indoor collected packets were used. This set was split into two subsets, training and validation. The first subset was used to train the IDS, as it learns the hidden patterns of both benign and malign AVTP packets. Once the IDS is fully trained, its performance is evaluated using the validation subset. In order to make the model less dependent on the data, a stratified 5-fold cross-validation process was used, which splits the dataset into 5 different folds, each one containing a training and validation subset with a proportional amount of benign and malign samples. For each fold, the model that presented the best overall performance regarding the evaluation metrics was saved to be further used in the test phase.

In the test phase, all the 5 best models that were obtained in the training phase are evaluated using the evaluation metrics on the test set. An overall view of our methodology is presented in Figure 26.

Figure 26 – Methodology of training and evaluation proposed multi-criteria optimized automotive Ethernet IDS.



Source: The author (2024)

5.3 RESULTS AND DISCUSSION

In this section, we present and discuss the detection results, detection time, and storage requirements of our proposed IDS, while also comparing it to two state-of-the-art automotive Ethernet IDSs.

5.3.1 Detection results

To evaluate the detection results of our IDS, we evaluated its accuracy, recall, precision, F1-score, and ROC AUC values. At first, the validation set was used to perform hyperparameter tuning. The only hyperparameter that needed to be tuned was the boundary, which controls the weights' initialization variance and plays a significant role in model convergence while using the LilNetX framework. The boundary value that provided the best results was 0.55. The validation results are presented in Table 5.

Table 5 – Detection-related metrics of the k-fold cross-validation for the indoors (training/validation) set.

Fold	Accuracy	Precision	Recall	F1-score	ROC AUC
0	0.9861	0.9832	0.9711	0.9770	0.9871
1	0.9883	0.9840	0.9779	0.9808	0.9902
2	0.9632	0.9698	0.9080	0.9375	0.9621
3	0.9867	0.9831	0.9733	0.9780	0.9902
4	0.9826	0.9811	0.9618	0.9712	0.9822
Mean	0.9814	0.9802	0.9584	0.9689	0.9823

Source: The author (2024)

As seen in Table 5, 4 of 5 folds presented F1-score values greater than 0.97, which shows

that the model obtained good results for the data available in the validation set. The mean value of the results was dragged down due to the fold number 2 results. Fold number 2 had the lowest recall value among all folds. We believe the reason behind this is that for this specific fold, there is a small amount of injected samples between the grouped samples, making it easier to confuse it with a benign sample and increasing the false negatives rate.

For the test set, we have obtained F1-Score values greater than the ones in the validation set. We credit this improvement to the correct detection of the malicious frames in the outdoor environments since the injected packets are the same as used in the model training and contain only 36 possible packets. We present our test set detection results in Table 6 alongside their comparison with two other works.

Moving on to the comparison with other state-of-the-art automotive Ethernet IDSs, the first IDS we compared our results with was introduced in (JEONG et al., 2021) and is referred to as 2D-CNN. The second IDS was proposed in (CARMO et al., 2022) and is referred to as XGB-ML, which uses an ensemble tree-based machine learning model and benefits from the low detection time of tree-based models.

Table 6 – Detection-related metrics for the outdoors (test) dataset.

Method	Accuracy	Precision	Recall	F1-score	ROC AUC
Our work	0.9913	0.9698	0.9884	0.9788	0.9974
2D-CNN	**0.9919**	0.9637	**0.9979**	**0.9805**	**0.9989**
XGB-ML	0.9747	**0.9727**	0.9357	0.9538	0.9805

Source: The author (2024)

Table 6 compares our results with the aforementioned works. When compared to the 2D-CNN results, we have obtained results that slightly differ only in the third or fourth decimal digit for most of the metrics. We credit this insignificant difference to using a convolutional neural network as our reference architecture. It is well known that CNN architectures are suitable for finding spatial relations in the data as its mainly used for image processing tasks. The XGB-ML method presented the worst results in the detection-related metrics for 4 out of 5 metrics of the test set, mainly because the XGB model is unsuitable for detecting spatial relations in the data, as it was initially developed to be used with tabular data.

5.3.2 Storage size

To compute the storage size of our proposed IDS, the number of bytes necessary to store the resulting network weights was used. The storage size was optimized interactively during training. By its end, the storage size of the model that presented a minimal drop in detection metrics was 11.7 KB. This value represents an improvement of approximately 900x if compared to the model proposed in (JEONG et al., 2021). A possible reason for this improvement is that the network weights distribution has a low standard deviation, making most of their values concentrated in a short range, enabling their representation with a small number of bits without damaging its detection results. The reduction in the number of filters also has a direct impact on the model storage size since it has fewer weights that need to be stored. A comparison of the obtained results is presented in Table 7. The XGB-ML method's storage size was obtained by saving the model as a `.pkl` and measuring the KBs needed to store the file.

The obtained storage size indicates that our IDS could be easily stored in memory-constrained devices, such as cheap microcontrollers, which usually have less than 1 MB of available storage. This result indicates that the amount of bits used to represent the network's weights is a crucial point to deploying DL-based IDSs in constrained-resources ECUs.

Table 7 – Storage size metrics comparison.

Method	**Storage size (KB)**
Our work	11.7
2D-CNN	10617.0
XGB-ML	10600.0

Source: The author (2024)

Table 8 – Detection time metrics comparison.

Method	**Detection time (µs/sample)**
Our work	1589
2D-CNN	2273
XGB-ML	250

Source: The author (2024)

5.3.3 Detection time

We computed the detection time as the mean time taken for the IDS to process a sample, i.e., we measured the total execution time to process a batch of samples and divided it by the number of samples in the batch. For the CPU used in our experiments, we have obtained a mean detection time of 1589 µs/sample. This result is less than real-world examples of the time between packet intervals of 3,157 and 1,735 µs/sample mentioned in (JEONG et al., 2021), showing that we can detect anomalies before a new packet arrives.

A comparison between our detection time results and the two other state-of-the-art methods is presented in Table 8. We have improved compared to the 2D-CNN method, primarily because of the optimized model's reduced number of necessary computations. We have removed the unnecessary computations from our final model. Specifically, the first convolutional layer proposed in (JEONG et al., 2021) (and presented in Table 4) was reduced from 32 to 27 output channels, while the second layer was decreased from 64 to 26 output channels. As this second layer serves as the input of the fully connected layer, the number of units in this last layer was also reduced from 20416 to 8294 units.

The difference between the XGB-ML model results is due mainly to the difference between the computations performed by each model. Where our model relies on matrix multiplications, the XGB-ML is based primarily on comparisons, a much less time-consuming computation.

5.3.4 Trade-off analysis

Finally, we analyzed the trade-off between the three evaluated DL-based IDSs for automotive Ethernet. Table 9 summarizes their F1-score, detection time, and storage size. The detection time and storage size in Table 9 were obtained by reproducing the works from both (JEONG et al., 2021; CARMO et al., 2022). Our proposed IDS provided well-balanced trade-off metrics, especially regarding storage size. Even with the difference in the detection in comparison to XGB-ML, we can still detect a cyberattack before a packet is received.

Table 9 – Trade-off analysis between detection time, storage size, and F1-score. The F1-score was obtained by considering the mean value of the test set evaluation for each work.

Method	Detection time (µs/sample)	Storage size (KB)	F1-score
Our work	1589	**11.7**	0.9788
2D-CNN	2273	10617.0	**0.9805**
XGB-ML	**250**	10600.0	0.9538

Source: The author (2024)

We have optimized both model storage size and detection time with a minimal drop of 0.0017 points in the F1-score. This storage size result means the model could be potentially stored in a simple microcontroller such as an RP2040 with only 264 KB of internal RAM and costs less than $1.

Although the XGB-ML model obtained a significant result regarding the detection time, its large storage size could be related to its ensemble method characteristic. It uses many simpler

models to compose a more robust model, which means the storage size increases with the number of simpler models.

At last, maintaining a high value of the F1-score is extremely important when working with safety-critical systems such as vehicles, where a misdetection may lead to an accident. We credit our minimal drop of F1-Score to using the 2D-CNN as our reference architecture, which has a higher capability of detecting and modeling complex data, such as network traffic containing images in their payloads.

6 MULTI-STAGE DEEP LEARNING-BASED INTRUSION DETECTION SYSTEM FOR AUTOMOTIVE ETHERNET NETWORKS

This chapter contains sections three to six of the paper "Multi-Stage Deep Learning-Based Intrusion Detection System for Automotive Ethernet Networks"(LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2024), which was submitted and published in a special issue of Ad Hoc Networks Journal of the best papers of SBRC 2023.

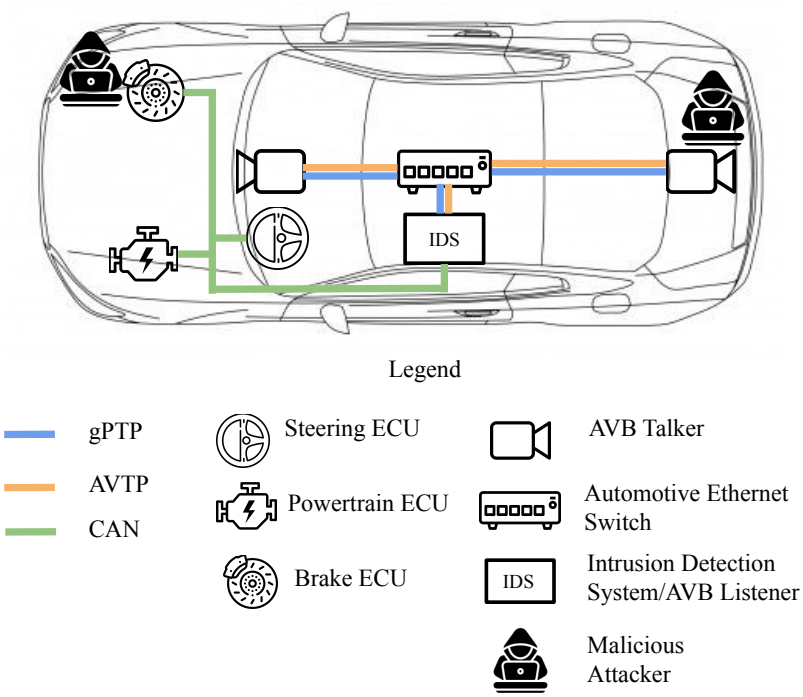
6.1 THREAT MODEL

In the threat model, we assume an attacker can access the IVN and send malicious frames through compromised AVTP talker and CAN nodes. This is illustrated in Fig. 27, where we also present an IVN architecture that uses two AVB talkers as part of the vehicle perception system that sends video streams using the AVTP protocol to an automotive Ethernet switch, which redistributes the streams to an AVB listener. The AVB listener decodes the streams and converts them to control signals sent over the CAN bus to the vehicle's main systems, such as brake, powertrain, and steering. The proposed IDS is deployed at the AVB listener because it can listen to the messages from the car's IVN (automotive Ethernet and CAN). It is also important to notice that the AVB talkers and listeners use the gPTP protocol to synchronize their clocks and guarantee that the information is sent and received at the right time.

Our work considers the publicly available state-of-the-art automotive Ethernet cyberattacks to be up-to-date with the most recent threats and to establish a baseline comparison with the works mentioned in Section 4. The following paragraphs explain the covered cyberattacks, their procedures, and their impact.

Injection attack. The working principle of an injection attack involves inserting harmful frames into a network to confuse legitimate nodes with false data, as illustrated in the malicious packets injection of Fig. 28a. The impact of such an attack goes from misguiding a camera-based parking assistance system to crashing into another vehicle in a parking lot to deceiving the decision-making system of an autonomous vehicle on a roadway, as mentioned in (MILLER; VALASEK, 2015), where the attackers injected a stop engine command with the vehicle in a highway.

Figure 27 – In-vehicle network architecture depicting its protocols, components, and compromised nodes.



Source: The author (2024)

Replay attack. In Fig. 28a, we present the working principle of the replay attack for an arbitrary IVN. It is worth mentioning that the capture and injection process depends on the IVN network topology and technologies. The replay attack is a specific injection attack where pre-captured frames or packets are injected into the network in a different context or time slot. The main goal of such an attack is to deceive the network nodes that rely on the replayed information. The first instance of a replay attack in an automotive Ethernet environment was presented in (JEONG et al., 2021). It highlights the potential impact of such an attack, which could be fatal if the replayed data is vital for the vehicle's perception system.

The aforementioned explanation is extended to the CAN replay attack, but with the consideration that the frames that will be replayed will come from previous CAN bus traffic. Other works have also considered the CAN replay attack for a CAN network, such as (Freitas De Araujo-Filho et al., 2021), but (HAN; KWAK; KIM, 2023) were the first to bring CAN packets into an automotive Ethernet network by encapsulating it in UDP packets.

MAC flooding attack. Fig. 28b depicts the process of a MAC flooding attack. This attack exploits the working principle of network switches. The process starts with the malicious node sending multiple messages with arbitrary MAC addresses, intending to fill the available

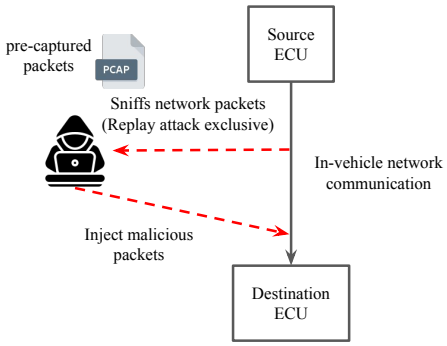
entries of the network switch's MAC table, which stores known MAC addresses common to the network (DAŞ; KARABADE; TUNA, 2015). Once the MAC table is full, the switch operates as a hub and broadcasts the received messages to the network nodes. This attack was first brought to automotive Ethernet networks by (HAN; KWAK; KIM, 2023).

PTP sync attack. The gPTP protocol synchronizes the clocks of timeTransmitter and timeReceiver nodes in a network to ensure that network messages meet timing requirements. These synchronizations are done periodically through sync and follow-up messages from the timeTransmitter node (VAL et al., 2022). The PTP sync attack is presented in Fig. 28c and consists of inducing a delay in the synchronization process by flooding the time information of the sync message, preventing nodes from synchronizing their clocks effectively (MOUSSA et al., 2019).

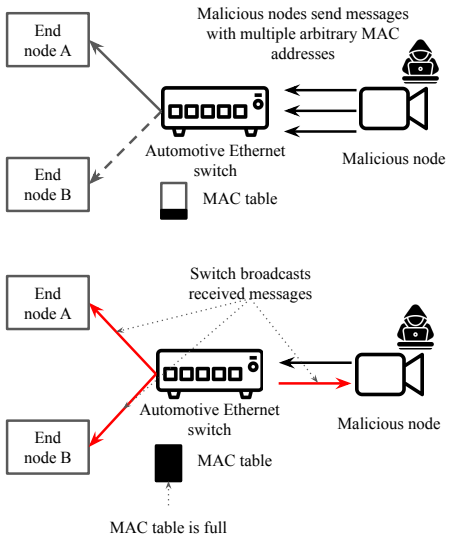
CAN denial of service attack. In the CAN bus arbitration, messages with lower IDs have higher priority (JO; CHOI, 2021). This concept is crucial to understanding the CAN DoS attacks, which involve flooding the bus with high-priority messages to block legitimate communication between nodes. This attack is widely known and available in different CAN bus datasets, such as (LAMPE; MENG, 2023; VERMA et al., 2022; SONG; WOO; KIM, 2020; SEO; SONG; KIM, 2018). The CAN replay attack was brought to an automotive Ethernet network by (HAN; KWAK; KIM, 2023) by tunneling the CAN frames in UDP. Despite the significant impact of DoS in the IVN, it can be easily detected by knowing the valid CAN IDs and blocking messages from unknown IDs. In Fig. 28d, we visually represent the CAN DoS attack, where a compromised node floods the CAN bus with multiple high-priority messages and disables the communication between the ECUs.

At last, we present in Table 10 a summary of the previously described cyberattacks, the IVN protocols they aim to compromise, and their impacts.

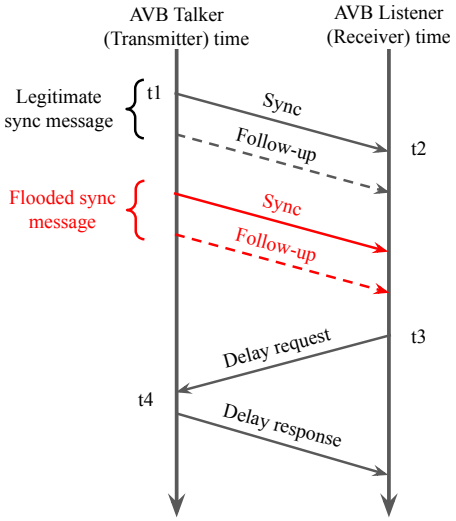
Figure 28 – Block diagram for some of the considered in-vehicle network cyberattacks for this work.



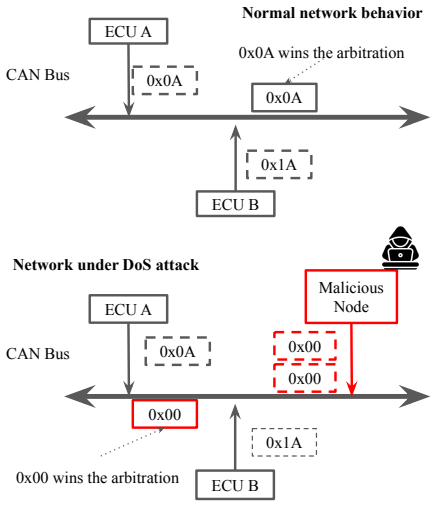
(a) Replay and injection attacks for an arbitrary in-vehicle network.



(b) MAC flooding attack.



(c) PTP sync attack.



(d) CAN DoS attack.

Source: The author (2024)

Table 10 – Table summarizes the existing attacks for automotive Ethernet and their impacts.

Attack	Protocol	Impact
Replay attack	AVTP	Replays pre-captured packets to compromise the legitimate nodes
Injection attack	AVTP	Injects unrealistic packets to deceive the network nodes
MAC flooding	AVTP	Overflow the MAC table until every packet is sent to all nodes
PTP sync attack	gPTP	Compromise the time synchronization process, delaying the transmission of time-sensitive packets
CAN DoS attack	CAN	Floods the network with high priority message, blocking the communication between legitimate nodes
CAN replay attack	CAN	Sends out-of-order/out-of-context data to misguide the legitimate nodes

Source: The author (2024)

6.2 PROPOSED IDS ARCHITECTURE

This section describes our proposed IDS architecture, including its modeling criteria and the specification of its main components. Fig. 29 presents a block diagram of our proposed IDS major components: the feature extractor, the attack detector, and the attack classifier stages.

Our proposed detection process is presented in Algorithm 2. It initially groups an amount of w_size + 1 network raw packets, where w_size represents the number of features that will be used as input for our IDS. The w_size value was 44, as proposed in (JEONG et al., 2021) via hyperparameter tuning targeting a high accuracy and F1-score, and a fast training time and validated in our previous work (LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023). Once the data is gathered, the detection process consists of a feature extractor stage and two detection stages that run in parallel. The feature extractor is responsible for extracting information from the grouped network raw packets, which are then passed on to the subsequent stages. The attack detector stage quickly determines if the received packets are normal or malicious and raises an alert if it finds any malicious packets. The attack classifier stage then determines the packet group's final classification and identifies the attack type if the data is found to be malicious. It is worth noting that the attack classifier of the proposed IDS is designed to be more accurate than the attack detector. Therefore, we have decided to use the output of the attack classifier as the final decision for the IDS. However, the attack classifier having the final decision does not exclude the need for the attack detector since the attack detector's primary goal is to reduce the detection time whenever possible. In cases where the output of the attack classifier is the same as that of the attack detector, the information on the attack classification is included in the IDS output. However, when the output of the attack classifier differs from the attack detector, the IDS decision is updated to match the attack classifier output.

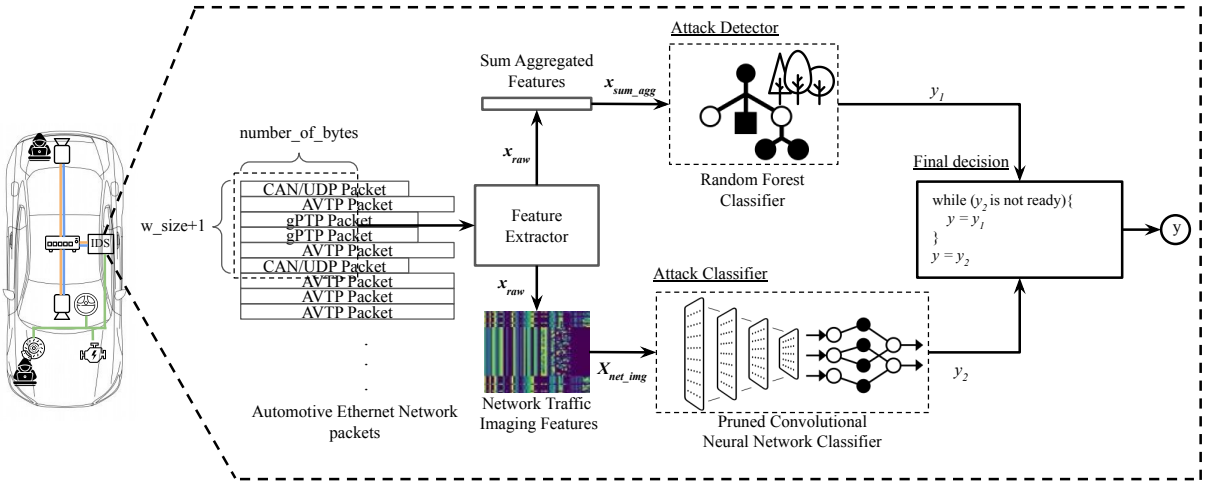
Once the detection process is concluded, the grouped raw packets are updated with w_slide packets, where w_slide represents the number of new packets added to the previously gathered packets, and the oldest w_slide are discarded from the group, following a moving window approach.

We propose the use of a multi-stage approach, where each stage has specific responsibilities that are directly related to the main demands for automotive IDSs: fast detection time and high accuracy with low false positives (SUN; YU; ZHANG, 2022; JO; CHOI, 2021; WU et al., 2020).

Algorithm 2 Proposed Multi-Stage Deep Learning based IDS detection process

w_size: Window size = 44
w_slide: Window slide = 1
$\boldsymbol{x}_{raw}$: Network raw packets vector
$\boldsymbol{x}_{sum\_agg}$: Sum aggregated features
$\boldsymbol{X}_{net\_img}$: Network traffic imaging features
y_1: One class output of IDS attack detector
y_2: Multi-class output of IDS attack classifier
y: Final classification
while The automotive Ethernet switch receives multi-protocol packets **do**
  group (w_size + 1) sequential packets in $\boldsymbol{x}_{raw}$
  if $\boldsymbol{x}_{raw}$ is full **then**
    $\boldsymbol{x}_{sum\_agg}$, $\boldsymbol{X}_{net\_img}$ <- FeatExtractor($\boldsymbol{x}_{raw}$)
    /* Stages are executed concurrently */
    y_1 <- AttackDetector($\boldsymbol{x}_{sum\_agg}$)
    y_2 <- AttackClassifier($\boldsymbol{X}_{net\_img}$)
    while AttackClassifier is running **do**
      y <- y_1
    end while
    y <- y_2
  end if
  Update $\boldsymbol{x}_{raw}$ with w_slide new packets and discard the latter w_slide
end while

Figure 29 – Block diagram of our proposed IDS main components.



Source: The author (2024)

The attack detector stage prioritizes a fast detection of potential cyberattacks, as shown in Eq. (6.1):

$$\mathrm{DT}_{\mathrm{attackdetector}} << \mathrm{DT}_{\mathrm{attackclassifier}}, \tag{6.1}$$

where $\mathrm{DT}_{\mathrm{attackdetector}}$ and $\mathrm{DT}_{\mathrm{attackclassifier}}$ are the detection times of the attack detector and attack classifier stages, respectively.

According to the abovementioned requirements, the desirable characteristics of the attack detector stage model are simplicity and faster inference time compared with deep neural network models. For this reason, we use the Random Forest model. This decision considers the model's simplicity and the previous results of tree-based models in detecting cyberattacks in automotive networks (YANG et al., 2019; Freitas De Araujo-Filho et al., 2021). Other supervised machine learning algorithms, such as Support Vector Machines (SVM) and XGBoost, could be considered for the attack detector model since both of them also provide a fast inference time without the need for specific hardware like GPU. However, XGBoost uses a boosting approach, where trees are built sequentially, which may result in longer inference times. On the other hand, SVM classifies the samples using a hyperplane, which usually requires a high number of computations for high-dimensional spaces, such as those from automotive Ethernet network environments.

On the other hand, the attack classifier stage is primarily responsible for detecting cyberattacks with higher detection metrics than the attack detector, as represented in Eq. (6.2):

$$\mathrm{DM}_{\mathrm{attackclassifier}} > \mathrm{DM}_{\mathrm{attackdetector}} \tag{6.2}$$

where $\mathrm{DM}_{\mathrm{attackclassifier}}$ and $\mathrm{DM}_{\mathrm{attackdetector}}$ represent the overall detection metrics of the attack classifier and attack detector stages, respectively. The detection metrics are later described in 6.3.3.

To fulfill the higher detection metrics requirement without the need for an extremely fast detection time, we have chosen to use the resulting model of our previous work (LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023), referred to as the Pruned Convolution Neural Network intrusion detection system (Pruned CNN IDS). The motivation of this choice is mostly regarding its previously obtained detection results, alongside its fewer parameters when

considered to the IDS proposed in (JEONG et al., 2021) and also the ease of expansion to a multi-class problem, when needed. Similar to the attack detector, other DL architectures could be considered for the attack classifier. Some possible architectures are the encoder and point mapper presented in (JEONG et al., 2023), and the CNN presented in (JEONG et al., 2021). Other widely known architectures in the image classification fields, such as the ResNet, could be evaluated. However, in recent work, different architectures (JEONG et al., 2023; JEONG et al., 2021) have obtained similar results (with detection metrics such as F1-Score and Accuracy of more than 99%). For this reason, we chose to use our proposed Pruned CNN architecture, which can maintain the performance results of other state-of-the-art while having a lower number of parameters and a lower storage size, which contributes to a lower detection time and further deployment in memory-constrained environments.

6.2.1 Feature extractor

Our proposed feature extractor is an expanded version of the feature generator initially proposed in (LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023) to handle the network packets from different protocols and provide more suitable input for the Random Forest classifier used as our first-stage detector.

The input of the feature extractor is a group of w_size + 1 packets from different protocols. We select a group of packets to identify temporal relations in the network packet data. The reason for the extra packet is that in a further step, we compute the sequential difference between the packets, resulting in a group of w_size features. This input can be represented as the vector $\boldsymbol{x_{raw}}$ presented in Eq. (6.3):

$$\boldsymbol{x_{raw}} = \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{\text{w\_size}} \end{bmatrix}, \tag{6.3}$$

where each $\boldsymbol{x}_i$, $0 \leq i \leq$ w_size, is also a vector whose length, referred to as packet_bytes, varies according to the in-vehicle network protocol and is equal to the number of bytes in the

packet. For example, if $\boldsymbol{x}_0$ is a packet from gPTP protocol, its packet_bytes could be between 60 and 90.

The second step in our proposed feature extractor is to select a specific amount of n_bytes in every packet in $\boldsymbol{x_{raw}}$ that will be further used as input of our IDS. In cases where packet_bytes is less than the specified n_bytes, we fill the remaining bytes with zero padding. We can then build the matrix $\boldsymbol{X_{padded}}$, that has a dimension of w_size+1 $\times$ n_bytes, and contains the selected amount of n_bytes from the raw network packets:

$$\boldsymbol{X_{padded}} = \begin{bmatrix} a_{00} & a_{01} & \dots & a_{0\text{n\_bytes}} \\ a_{10} & a_{11} & \dots & a_{1\text{n\_bytes}} \\ \vdots & \vdots & \ddots & \vdots \\ a_{\text{w\_size}0} & a_{\text{w\_size}1} & \dots & a_{\text{w\_sizen\_bytes}}, \end{bmatrix} \tag{6.4}$$

whose elements $a_{ij}$are defined by Eq. (6.5):

$$a_{ij} = \begin{cases} x_{ij}, & \text{if } 0 \leq j \leq \text{packet\_bytes} \\ 0, & \text{if packet\_bytes} < j \leq \text{n\_bytes} \end{cases} \tag{6.5}$$

where x_{ij} are the elements of $\boldsymbol{x_i}$.

To capture the variations of the network packet fields over time, we take the modulus of the difference between the bytes of consecutive samples to form the $\boldsymbol{X_{diff}}$ matrix composed of b_{ij} elements, as shown in Eq. (6.6)

$$b_{ij} = \mod \left(\frac{a_{i+1j} - a_{ij}}{255} \right), \tag{6.6}$$

where $0 \leq i \leq$ w_size - 1 and $0 \leq j \leq$ n_bytes $- 1$.

To form the network traffic imaging feature, the last step consists of splitting the bytes of $\boldsymbol{X_{diff}}$ into nibbles, forming the $\boldsymbol{X_{net\_img}}$ matrix, comprised of n_{ij} elements, as depicted in Eq. (6.7)

$$n_{ij} = \begin{cases} b_{i((j+1)/2)} >> 4, & \text{if } j \text{ is odd} \\ b_{i((j+1)/2)} \text{bitwise and 0x0F}, & \text{if } j \text{ is even} \end{cases}, \tag{6.7}$$

for $0 \leq i \leq$ w_size - 1 and $0 \leq j \leq 2 \cdot$ n_bytes.

The resulting $\boldsymbol{X_{net\_img}}$ can identify the spatial-temporal relation of the network data, and it will serve as input of our attack classifier model, the Pruned CNN IDS, because of its ability to handle and potentialize the information contained in two-dimensional data.

However, generating the input features for our attack detector model is still necessary. Since we have chosen a Random Forest model, its most suitable input is a one-dimensional vector. Therefore, we have chosen to aggregate the $\boldsymbol{X_{net\_img}}$ with a column-wise sum to create the needed one-dimensional vector that enhances the differences between the fields of the grouped network packets. We refer to the resulting vector as $\boldsymbol{x_{sum\_agg}}$, that is composed of c_i, which expression is presented in Eq. (6.8)

$$c_i = \sum_{j=0}^{\text{w\_size-1}} b_{ij}. \tag{6.8}$$

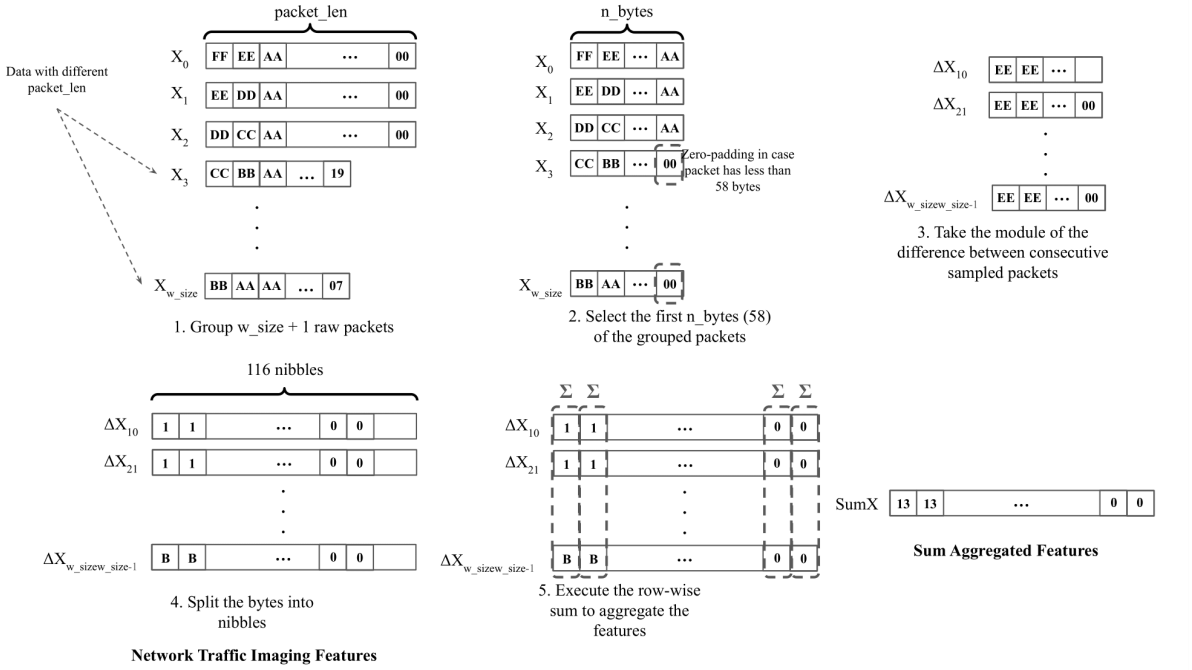
In Fig. 30, we present a visual representation of our Feature extractor steps for the case where w_size is equal to 44, and n_bytes is equal to 58. The values chosen are consistent with our previous work (LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023) and can be easily adapted for other network applications. These values encompass the contained information in a PTP Sync frame and the information up to the first two bytes of the PTP Follow_up frame. For AVTP packets, we cover information up to the first two bytes of the ISO/IEC 13818-1 frame. This includes vital information such as part of the video data and source and destination addresses. Lastly, we cover the CAN ID and payload information for cases where a CAN frame is encoded in a UDP packet. This is because the maximum length of CAN frames in the dataset from (HAN; KWAK; KIM, 2023) is 9 bytes, and the CAN payload is stored before the 58th byte of the UDP frame.

6.2.2 Attack detector stage: Random Forest classifier

The Random Forest classifier is a machine learning technique that combines multiple decision tree models, considered weak learners because they only consider a small number of features during training. The output of each tree is then used to determine the final decision of the Random Forest model (BREIMAN, 2001). In the attack detector of our proposed IDS, this algorithm identifies whether a group of packets is normal or malicious, making it a one-class classifier.

Several hyperparameters can be tuned to achieve better results using the Random Forest

Figure 30 – Steps of the proposed feature extractor. It is important to notice that this feature extractor has two outputs: the network traffic imaging features that will be fed to the Pruned CNN model and the sum aggregated features that the Random Forest model will use.



Source: The author (2024)

classifier. We have chosen to variate only three of the available hyperparameters, which are the most related to the amount of information the Random Forest model will use to make its decisions. Their description is presented below:

- n_estimators: The number of decision tree estimators used in the Random Forest.
- max_depth: The maximum depth the decision tree models can achieve. Increasing this parameter may lead the model to overfit.
- max_features: The maximum number of features that can be used to train the decision tree models.

At last, in Table 11, we present the final values for the mentioned hyperparameters for the first-stage models in the AEID and TOW-IDS datasets. These values were obtained through hyperparameter optimization for each dataset. We have considered values between 50-300 for the n_estimators because the detection time increases with the n_estimators, and our main criterion was a short detection time. For the max_depth and max_features, we have evaluated values between 3-10. The reason behind these values is that weak learners become more prone to overfitting as long as they gain more depth and consider more features.

Table 11 – The Random Forest models hyperparameters used for each dataset.

Hyperparameter	Dataset	
	AEID	TOW-IDS
n_estimators	100	200
max_depth	4	10
max_features	3	7

Source: The author (2024)

6.2.3 Attack classifier stage: Pruned Convolutional Neural Network classifier

For the attack classifier model of our proposed IDS, we have chosen to use the Pruned CNN architecture from our previous work (LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023), which is the result of pruning and quantization of the CNN architecture proposed in (JEONG et al., 2021). We have used the multi-criteria pruning and quantization technique proposed in (GIRISH et al., 2022), which uses a training loss composed of three major components: detection rate, storage size, and detection time. This technique induces a low-bit representation of weights by imposing an entropy penalty, which also contributes to the model sparsity by setting several network parameters to zero, making them susceptible to pruning. It is important to mention that the optimization is performed during the training phase and updates the network weights iteratively based on the multi-criteria loss function without needing a post-training step. This decision was supported by the obtained results of the quantized and pruned CNN architecture, which showed a reduction of 900x in the storage size and a speedup of 1.4x when compared to the results obtained (JEONG et al., 2021). Moreover, the detection metrics showed only a slight drop, with the highest difference in the second decimal digit.

We have only utilized the resulting Pruned CNN architecture obtained in (LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023), which includes the number of channels and units in the convolutional and linear layers of the model. Using the same quantized weights obtained before was unnecessary, as we now have an attack detector stage focusing on lower detection time.

Moreover, we have also adapted the Pruned CNN model in terms of its number of outputs. Since the TOW-IDS dataset (HAN; KWAK; KIM, 2023) is a multi-label dataset, we now have a multi-label classification problem. Therefore, the output layer of the network must contemplate the number of labels present in the dataset.

The layer types, activation function, dropouts, regularization, and hyperparameters considered for the attack classifier model for both datasets are presented in Table 12. The only

dataset-dependant hyperparameter is out_feat, which depends on the number of cyberattacks present in the considered dataset, and its expression is presented in Eq. (6.9). For the training process, we have used a batch size of 64, 30 epochs with an early stopping patience of 5 epochs, Adam optimizer, and a learning rate of 0.001.

Table 12 – Pruned CNN architecture and hyperparameters obtained in (LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023).

Layer name	Activation	Regularization	Hyperparameters
Conv2D_1	ReLU	Batch Norm	in_ch=1, out_ch=27, kernel_size=5, stride=1, padding='same'
MaxPool_1	-	-	kernel_size=2, stride=2
Conv2D_2	ReLU	Batch Norm	in_ch=27, out_ch=26, kernel_size=5, stride=1, padding='same'
MaxPool_2	-	-	kernel_size=2, stride=2
Flatten	-	Dropout	in_feat=8294, out_feat=64
Dense	ReLU	Dropout	in_feat=64, out_feat=Eq. (6.9)
Output	Sigmoid	-	-

Source: The author (2024)

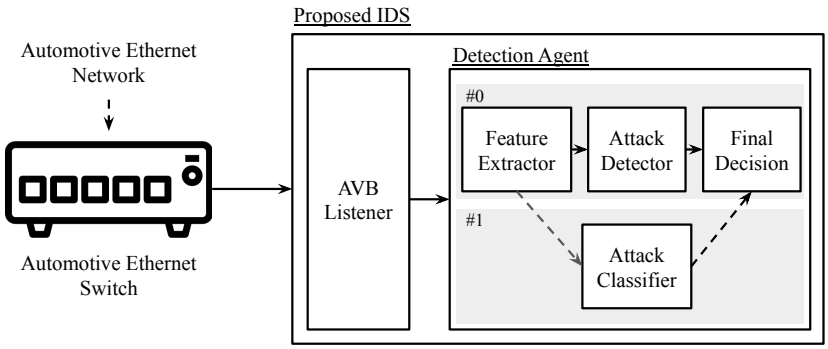
$$\text{out\_feat} = \begin{cases} 1, & \text{if dataset is AEID} \\ 6, & \text{if dataset is TOW-IDS} \end{cases}, \tag{6.9}$$

6.2.4 Proposed IDS deployment and update

In Fig. 31, we present an architecture to deploy our proposed IDS in a real automotive Ethernet environment. The architecture comprehends two major components: an AVB listener and a detection agent. The AVB listener collects the automotive Ethernet packets and sends them to the detection agent. The detection agent comprises the software components for the IDS decision-making. Since our IDS is based on a multi-stage approach, where each stage is independent, we recommend executing the stages concurrently.

Our proposed IDS is designed to be trained offline and installed in the vehicle. However, it allows regular updates to maintain its performance over time. These updates can be conducted periodically or by analyzing drops in a performance metric, such as false positives or false negatives. Once the need for an update is identified, a new offline training would be required, considering both old and new data.

Figure 31 – Proposed IDS deployment architecture.



Source: The author (2024)

6.3 METHODOLOGY AND EXPERIMENTAL EVALUATION

This section presents the automotive Ethernet datasets used throughout our experiments, methodology, experimental setup, and evaluation metrics used to assess the performance of our proposed IDS. We made our code available in our github repository to ease reproducing our experimental results.

We have chosen the Python programming language and the PyTorch frame due to their vast use, documentation, and community in developing machine learning and deep learning algorithms. The training, validation, and test experiments were conducted in an Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz CPU and in an NVIDIA GeForce RTX 3090 GPU. The timing metrics were measured using an Intel(R) Xeon(R) CPU @ 2.20GHz to compare our extended results with the results presented in (LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023).

6.3.1 Dataset presentation

The datasets used to evaluate our proposed IDS are the publicly available automotive Ethernet intrusion dataset (AEID) (JEONG et al., 2021) and the TOW-IDS dataset (HAN; KWAK; KIM, 2023). We have chosen to evaluate our proposed IDS in both datasets to show its generalization capability to work in several domains and also to be able to compare with the existing works regarding automotive Ethernet intrusion detection, such as (JEONG et al., 2021; CARMO et al., 2022; LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023; HAN; KWAK; KIM, 2023; SHIBLY et al., 2023).

The AEID dataset consists of genuine network traffic that was recorded using a camera

mounted on a vehicle. The captured video streams were sent over as AVTP packet payloads. The data was collected in both indoor and outdoor environments and is available in .pcap files. This dataset includes two types of data: normal traffic and injected data from a previously captured group of 36 packets, which were used to simulate a replay attack implementation.

The TOW-IDS dataset is also made available in .pcap files and contains data from five attack scenarios for different in-vehicle network protocols. The available attacks in this dataset are CAN DoS, CAN replay, frame injection, MAC flooding, and PTP sync attacks. The dataset already has a file containing one label per network packet, specifying the type of the attack in case it is an attack.

To train our proposed IDS in a supervised manner, we needed a labeled dataset suitable for our feature extractor process. Therefore, we established specific labeling criteria for dataset we used. The labeling criteria for the AEID dataset are outlined in Algorithm 3, and the labeling criteria for the TOW IDS dataset can be found in Algorithm 4.

Algorithm 3 AEID dataset labeling criteria

```
InjectedPacketsGroup: Set of 36 network packets containing injected frames
w_size: 44
RawGroupedPackets: Set of w_size networks packets without label
for RawGroupedPackets in AEID Dataset do
    if any(InjectedPacketsGroup) is in (RawGroupedPackets) then
        Y <- Attack
    end if
    Y <- Normal
end for
```

Algorithm 4 TOW-IDS dataset labeling criteria

```
GroupedLabels: Set of w_size labels
w_size: 44
for GroupedLabels in TOW-IDS Dataset do
    if NumOfAttacks > 1 then
        Y <- Most frequent attack in GroupedLabels
    else if NumOfAttacks == 1 then
        Y <- CurrentAttack
    else
        Y <- Normal
    end if
end for
```

For the AEID dataset, it was necessary to generate the labels based on the pre-captured set of injected network packets. The network packets contained in this file were considered

malicious. Therefore, if the grouped packets contained any injected packets, the grouped packets were considered malicious. The final class distribution for the AEID dataset is presented in Table 13.

Table 13 – Class distribution after labeling for AEID dataset.

Class	Train	Test
Normal	172,668 (26,84%)	1,350,550 (72,20%)
Replay Attack	470,596 (73,16%)	519,939 (27,80%)

Source: The author (2024)

Since the TOW-IDS dataset contains one label per sample, it was necessary to adapt the label to refer to a group of packets. The main difference in our labeling process is that if there is more than one attack in the packets, the most frequent attack inside the packets is considered the label for this specific group. Table 14 presents the final class distribution. Our distribution differs from the one presented in (HAN; KWAK; KIM, 2023) since the distribution shown in (HAN; KWAK; KIM, 2023) refers to the class distribution per sample without considering any grouping in the data. We believe this is a positive enforcement of our labeling criteria that helps us achieve more representative samples of the attacks. When considering a real scenario, it is essential to know if an attack is happening in the analyzed time window and not only if it just started to happen. The data distribution presented in Table 13 and Table 14 can be obtained by executing the `execute_feature_generator.py` script in our code repository using the raw datasets as input.

Table 14 – Classes distribution after labeling for TOW-IDS dataset.

Class	Train	Test
Normal	560,908 (46,60%)	443,336 (56,01%)
CAN DoS	178,710 (14,85%)	86,721 (10,96%)
CAN Replay	101,629 (8,44%)	103,090 (13,02%)
PTP Sync	193,380 (16,07%)	76,236 (9,63%)
Frame Injection	64,676 (5,37%)	31,437 (3,97%)
MAC Flooding	104,389 (8,67%)	50,746 (6,41%)

Source: The author (2024)

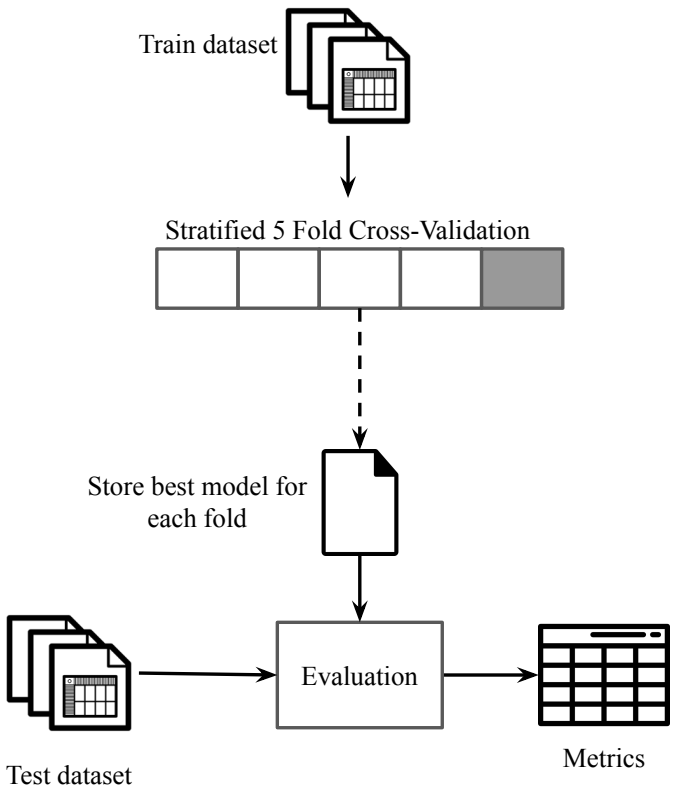
6.3.2 Experimental evaluation

The experimental evaluation of our IDS is divided into a training and a test phase. Both phases are conducted individually for both stages since each stage is independent and uses different features as inputs.

The training phase uses a stratified 5-fold cross-validation technique to make our IDS less dependent on data. This method splits the dataset into five folds, each containing the same proportion of the available classes. The model that presented the best overall performance for each fold was stored for further use in the test phase.

The test phase evaluates the stored models in the test dataset that have yet to be used to simulate how the IDS would perform in a real-world scenario with unseen data. Fig. 32 presents an overall view of our methodology. It is important to mention that the training and test phases were conducted two times for each dataset, one for the attack detector stage Random Forest classifier and another for the attack classifier stage Pruned CNN classifier.

Figure 32 – Methodology used for training, validating, and testing our proposed IDS.



Source: The author (2024)

6.3.3 Evaluation metrics

To evaluate the obtained results, we have considered metrics widely used in the literature for intrusion detection problems, alongside the detection time, which is used to assess how fast the intrusion detection would notice and report the occurrence of an attack. The metrics and the detection time are presented in equations (6.10)-(6.15):

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}, \tag{6.10}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{6.11}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{6.12}$$

$$\text{F1-score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}, \tag{6.13}$$

$$\text{ROC AUC} = \int_0^1 TPR(FPR)\, dFPR, \tag{6.14}$$

and

$$\text{DT} = \frac{\text{Mean detection time}}{\text{Batch size}} \left[\frac{\mu s}{sample}\right], \tag{6.15}$$

where TP is True Positive, TN is True Negative, FN is False Negative, FP is False Positive, TPR is True Positive Rate, FPR is False Positive Rate, ROC AUC is the Receiver Operating Characteristic Area Under Curve, and DT is the Detection Time.

These metrics are used to evaluate our IDS and consider essential aspects of its performance. Accuracy measures how well the model can predict the data. However, since our dataset is highly imbalanced, Accuracy should be used with other metrics. For this reason, we have also used the Precision, Recall, and F1-score metrics to evaluate how well the model detects normal data, anomaly data, and a mean value between them. We have considered using ROC AUC to assess how well our IDS performs by varying its detection threshold.

It is important to notice that Eq. (6.15) measures the individual stages detection time. Therefore, we can also formulate an equation to give us the result for our IDS overall detection time by combining the detection time from both stages and their Accuracy, as follows:

$$\begin{aligned} \mathrm{DT}_{\mathrm{overall}} = \mathrm{DT}_{\mathrm{attackdetector}} \cdot \mathrm{Acc}_{\mathrm{attackdetector}} + \\ \mathrm{DT}_{\mathrm{attackclassifier}} \cdot (\mathrm{Acc}_{\mathrm{attackclassifier}} - \mathrm{Acc}_{\mathrm{attackdetector}}), \end{aligned} \tag{6.16}$$

where $\mathrm{DT}_{\mathrm{attackdetector}}$ and $\mathrm{DT}_{\mathrm{attackclassifier}}$ are the detection time for the attack detector and attack classifier stage, respectively, and $\mathrm{Acc}_{\mathrm{attackclassifier}}$ and $\mathrm{Acc}_{\mathrm{attackclassifier}}$ the Accuracy for the attack detector and attack classifier stage, respectively.

6.4 RESULTS AND DISCUSSION

In our experiments, we evaluated the detection rate, confusion matrix, detection results, and detection time for both stages of our proposed IDS in both available datasets. The goal was to understand if our modeling criteria presented in Eqs. (6.2) and (6.1) were being properly satisfied. Finally, we compared our IDS to three state-of-the-art automotive Ethernet IDSs that used the AEID dataset (JEONG et al., 2021), (CARMO et al., 2022), and (LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023) and two other works that propose IDSs but using the TOW-IDS dataset (HAN; KWAK; KIM, 2023) and (SHIBLY et al., 2023).

6.4.1 Detection results

We used the confusion matrix to evaluate the number of true positives, false positives, true negatives, and false negatives in each stage of our proposed IDS. The confusion matrix can assist in a visual and quantitative analysis of the performance of our proposed IDS. For the AEID dataset, our attack detector stage detected 99.7% of the packets correctly, but it also presented a high false positive rate compared to the attack classifier stage. Moreover, in the attack classifier stage, only 270 samples of 1,870,489 were incorrectly classified, representing less than 2% of the entire dataset. It is interesting to understand that even though the attack detector stage was based on a simpler model and feature, it could still achieve significant detection results. The attack detector stage results show that a traditional machine learning-based IDS can provide fast detection time for attacks such as the replay attacks in the AEID dataset without sacrificing its high detection results. The confusion matrix for the attack detector and attack classifier stages of our proposed IDS for the AEID dataset are presented in Figs. 33a and 33b.

However, for the TOW-IDS dataset, it is possible to notice that the amount of samples incorrectly classified has risen compared to the AEID dataset results. One possible reason for the obtained results is the heterogeneity of attacks present in the dataset, which indicates that the sum-aggregated features need to be revisited or incremented to provide results as high as the ones obtained for the AEID dataset. Furthermore, moving on to the attack classifier stage results, we see that the attack classifier stage model could correctly classify most packet groups in the test set. The samples with more incorrect classifications were the normal ones, mostly due to their higher number of training samples than the other attacks. However, classifying normal samples as attacks represents false positives and is not as harmful as false negatives for IDSs. The confusion matrix for the attack detector and attack classifier stages of our proposed IDS for the TOW-IDS dataset can be seen in Figs. 33c and 33d, respectively.

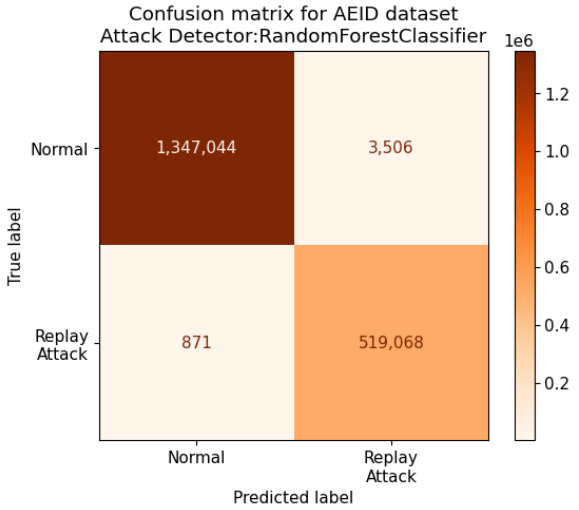
We have also used the ROC curve to analyze the TPR and FPR among the stages of our IDS. Each curve point represents a pair of TPR and FPR achieved for specific thresholds. This kind of analysis can assist in deciding the detection threshold to prioritize a higher true positive or false positive rate, depending on the application.

The ROC curve for our proposed IDS is presented in Fig. 34a. As can be seen, the thresholds did not significantly impact the results of both the attack detector and attack classifier stages; this is primarily due to the stages' capability to classify a normal sample directly as zero and all the other samples as one.

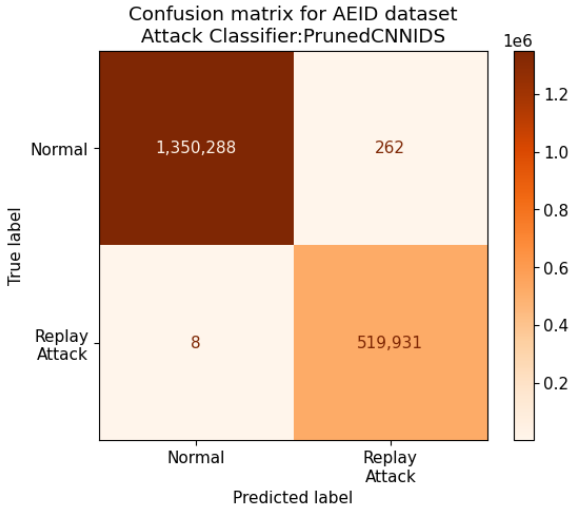
While for the TOW-IDS curves, presented in Fig. 34b, it is possible to see the performance decay in the attack detector stage model. As we previously discussed, this is mainly because the model cannot distinguish, as well as the attack classifier stage, the attacks present in the TOW-IDS dataset.

We compared our proposed IDS to the state-of-the-art works regarding their detection results: Accuracy, Precision, Recall, F1-Score, and ROC AUC values. For the AEID dataset, we have obtained the first and second-best results with our attack classifier and attack detector stage models, respectively. The comparison results with the works that used the AEID dataset are presented in Table 15. Interestingly, we obtained better results than those presented in (JEONG et al., 2021) with a model with fewer connections. This could be explained by the lottery-ticket hypothesis discussed in (FRANKLE; CARBIN, 2018), which states that the central information of a deep neural network resides primarily on a few connections. Additionally, our labeling criteria generated more attack samples for the training dataset than the one considered in (JEONG et al., 2021).

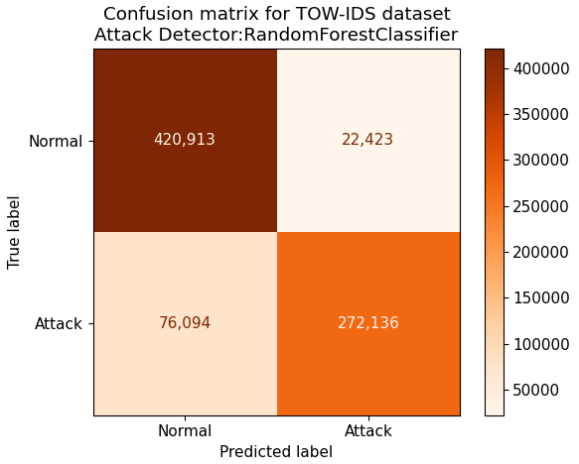
Figure 33 – Confusion matrix results for the evaluated datasets. C_D: CAN DoS Attack, C_R: CAN Replay Attack, P_I: PTP Sync Attack, F_I: Frame Injection Attack, M_F: MAC Flooding Attack.



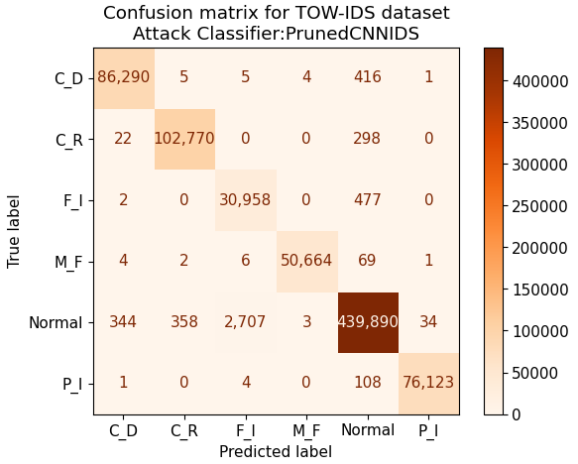
(a) Attack detector confusion matrix for AEID dataset.



(b) Attack classifier confusion matrix for AEID dataset.



(c) Attack detector confusion matrix for TOW-IDS dataset.

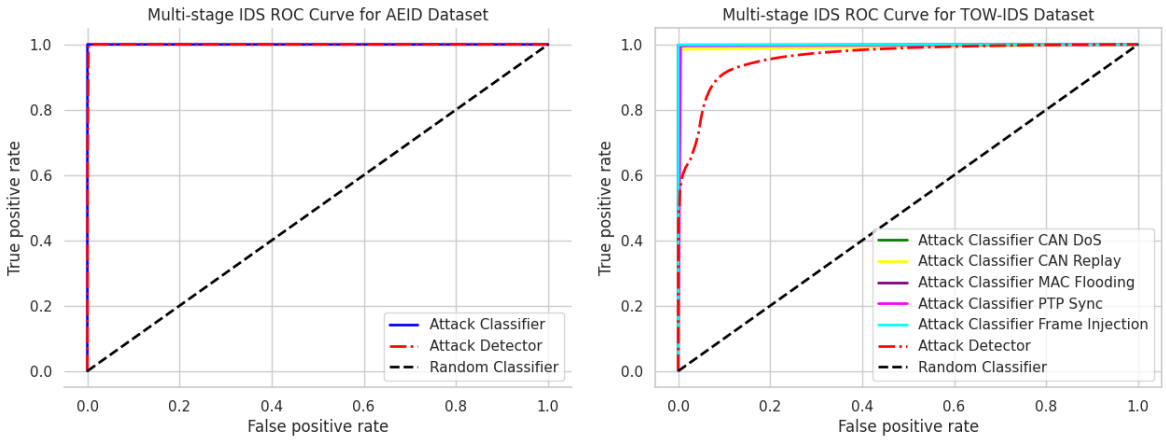


(d) Attack classifier confusion matrix for TOW-IDS dataset.

Source: The author (2024)

In Table 16, we present the detection results achieved by our proposed IDS for the TOW-IDS dataset. In this scenario, our attack classifier achieved the second-best detection results among the compared works, differing only in the third decimal results from the results presented in (HAN; KWAK; KIM, 2023). The attack classifier stage detection metrics for the TOW-IDS dataset show that our multi-stage technique provides the expected results since the attack classifier stage's primary goal is to achieve higher detection results. We noticed a performance drop in the attack detector stage when compared to the AEID dataset results. As a result, we have conducted an analysis of the attack detector results for each attack in the TOW-IDS dataset. This analysis aims to identify the possible reasons for the performance drop and

Figure 34 – ROC Curve of our proposed IDS stages in both evaluated datasets.



(a) ROC Curve of IDS stages for AEID dataset. (b) ROC Curve of IDS stages for TOW-IDS dataset.

Source: The author (2024)

Table 15 – Test set results for AEID dataset. The results from (JEONG et al., 2021) were obtained through code reproduction, while for (CARMO et al., 2022), we have used the results presented in the original paper.

Work	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Our work (Attack detector)	0.9976	0.9932	0.9982	0.9957	0.9993
Our work (Attack classifier)	**0.9997**	**0.9995**	**0.9997**	**0.9996**	**0.9998**
(LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023)	0.9913	0.9698	0.9884	0.9788	0.9974
(CARMO et al., 2022)	0.9747	0.9727	0.9357	0.9538	0.9805
(JEONG et al., 2021)	0.9919	0.9637	0.9979	0.9805	0.9989

Source: The author (2024)

provide additional information to support future work. In Table 17, we present the number of false and true negatives per attack for our attack detector stage in the TOW-IDS dataset. One can observe that the CAN Replay attack significantly impacts the performance results, mainly due to the number of false negatives it generates. However, the attack detector's performance drop regarding the CAN Replay attack is compensated by the improved detection results of our proposed attack classifier.

Finally, our accuracy modeling criteria defined in Eq. (6.2) were satisfied based on our experimental results. Surprisingly, for the AEID dataset, the difference in detection results between the attack detector and attack classifier stages was only on the third decimal digit. This shows that our attack detector stage can handle more straightforward attacks, such as the replay attacks present in the AEID dataset. Moreover, we have observed a difference in the first decimal digit between the attack detector and attack classifier stage models for the

Table 16 – Test set results for TOW-IDS dataset. We have used the results presented in the original paper for both (HAN; KWAK; KIM, 2023) and (SHIBLY et al., 2023) works.

Work	Accuracy	Precision	Recall	F1-Score
Our work (Attack detector)	0.8740	0.9238	0.7778	0.8446
Our work (Attack classifier)	0.9962	**0.9960**	**0.9962**	0.9960
(HAN; KWAK; KIM, 2023)	**0.9965**	-	-	**0.9974**
(SHIBLY et al., 2023)	0.9700	0.9400	0.9500	0.9500

Source: The author (2024)

Table 17 – Attack Detector Random Forest Classifier false negatives and true negatives per attack for the TOW-IDS dataset.

Attack	False Negatives (% Total)	True Negatives (% Total)
CAN DoS	6931 (9.11)	79790 (29.32)
CAN Replay	62593 (82.26)	40497 (14.88)
MAC Flooding	38 (0.05)	50708 (18.63)
PTP Sync	99 (0.13)	76137 (27.98)
Frame Injection	6433 (8.45)	25004 (9.19)
Total	76094 (100.00)	272136 (100.00)

Source: The author (2024)

TOW-IDS dataset. Although this result still fulfills our modeling criteria, it shows room for improvement in the attack detector stage model for more complex datasets and cyberattacks.

6.4.1.1 On the explainability of the attack detector on the TOW-IDS dataset

We have conducted an initial explainability analysis focusing on CAN replay attack samples to highlight possible directions and identify reasons for the detection performance reduction of our attack detector model in the TOW-IDS dataset. An explainability analysis aims to understand how information is used in the model's decision process. To enhance this understanding, it is crucial to have domain knowledge to identify which group of features should be the most important in each evaluated scenario. In this direction, Figure 35a presents the CAN over UDP frame of the TOW-IDS dataset, depicting the frame fields. Since our main goal is to identify possible reasons for the misdetection of CAN replay attacks in our attack detector model, the most important features should consider the fields of CAN frames, such as the CAN ID,

DLC, and payload, which, according to Figure 35a, are within the nibbles of indexes 72-75 and 84-101 (JEONG et al., 2023).

At first, we carried out a permutation feature importance analysis, which consists of permuting each feature value and measuring the impact of the permutation on the model performance, where the most important features significantly change the model results (BREIMAN, 2001). For this analysis, we considered normal and CAN replay attack samples from the test set.

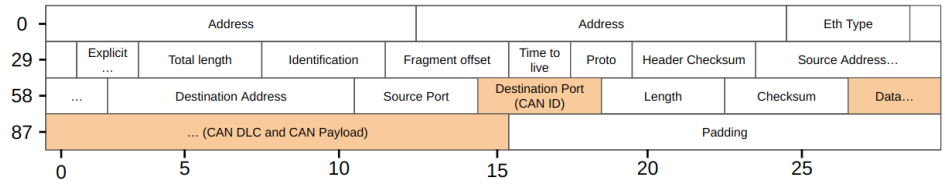
In Figures 35b and 35c, we present the permutation feature importance analysis for normal and CAN replay attack samples, where the brighter squares represent the most important features. For the normal samples, the most important feature was the nibble of index 93, and for the CAN replay attack, it was the nibble of index 98, where both nibbles are related to the CAN payload (see Fig. 35a). For the CAN replay attack samples, we can observe that the features of indexes 74 and 75, which are related to the CAN ID, obtained high importance, indicating that a variation in the CAN ID field may suggest a CAN replay attack. We can also observe an overlap of important features within the CAN payload in both normal and CAN replay attack samples, indicating that these features are important to both classes.

For our second explainability analysis, we have used the Trustee Framework (JACOBS et al., 2022b), which focuses on generating a high-fidelity and easy-to-interpret decision tree from the training data and a black box model. We have conducted this analysis only with normal and CAN replay attack samples from the training set and employed the same stratified 5-fold cross-validation approach to generate one surrogate decision tree per fold. The resulting high-fidelity trees had more than 100 branches, which was difficult to analyze. Therefore, we have considered the top-k pruned trees, with the default value of $k = 10$, to provide an easier-to-interpret tree. The pruned decision trees presented a minimal variation between the trees' decision thresholds and considered features.

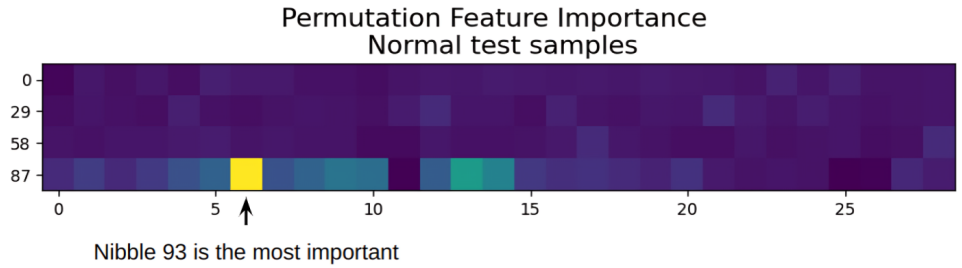
In Figure 36, we present one of the pruned decision trees from the Trustee framework. Based on the resulting decision tree, we can observe that the features considered for splitting the nodes are related to the CAN payload and ID fields, as shown by the previous permutation feature importance analysis (Figs. 35b and 35c). However, the split that uses feature 74 shows a high value of gini impurity, indicating a misclassification probability of approximately 40% in this branch.

The observed high gini impurity values indicate an uncertain decision boundary between normal and CAN replay attack samples, making it hard to find the best split. This result

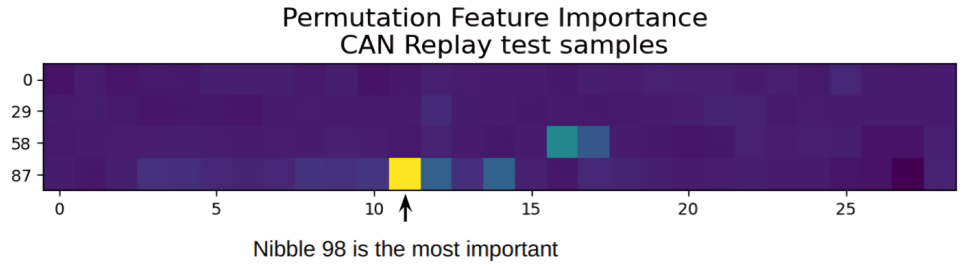
Figure 35 – CAN over UDP packet features and permutation feature importances for normal and CAN replay attack test samples. A brighter color indicates a higher importance.



(a) CAN over UDP packet features.



(b) Normal test samples permutation feature importance.



(c) CAN replay test samples permutation feature importance.

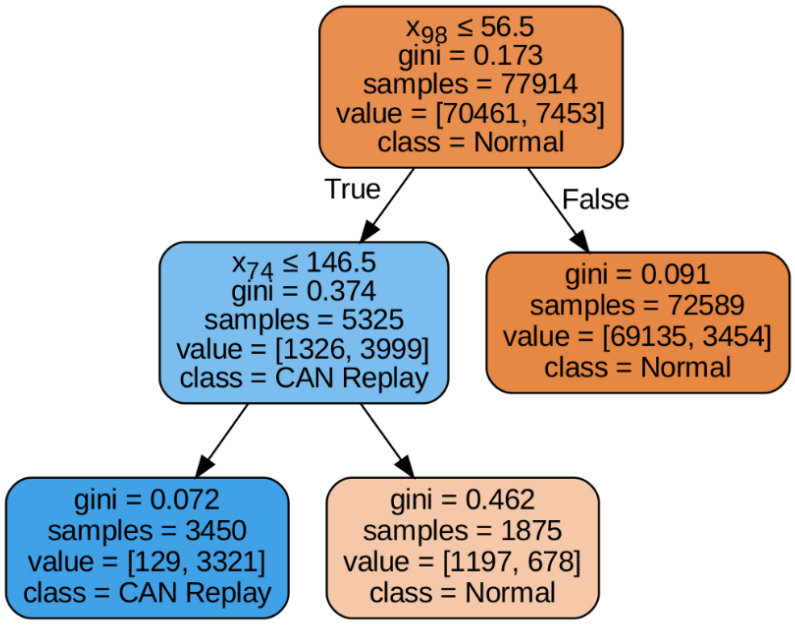
Source: The author (2024)

corroborates with the information that the CAN replay attack is more complex than other attacks as they have valid CAN IDs and payload values (JEONG et al., 2023; HAN; KWAK; KIM, 2023). Due to the higher complexity nature of the CAN replay attack and the TOW-IDS dataset heterogeneity, some factors may contribute to out-of-distribution (o.o.d) samples in the test set. These reasons include but are not limited to, a high cardinality in the values of the features coming from different CAN IDs and payloads. Another possible reason is the variable number of CAN frames in the sampling window considered for the feature extractor.

6.4.2 Detection time results

For the detection time results, we have used Eq. (6.15) with a batch size of 64 and considering the mean value of 500 samples. The same sample amount was used to measure

Figure 36 – Pruned decision tree obtained using Trustee framework with normal and CAN replay attack samples.



Source: The author (2024)

the time taken in the feature generation step.

Our detection time results are presented in Table 18. We have assessed the detection time results for both stages of our proposed IDS. As expected, the attack detector stage presented the best detection time results for AEID and TOW-IDS datasets, achieving 107 and 221 μs/sample, respectively. These results are due to Random Forest models' simple inference process computations. The IDS that obtained the second-best detection time in the compared works was proposed in (CARMO et al., 2022), also based on a Decision Tree ensemble technique, the XGBoost. At last, we could also compare our attack classifier stage results with those obtained in (JEONG et al., 2021). We were able to have a better performance, mainly because our CNN has fewer parameters than the one proposed in (JEONG et al., 2021).

The baseline works do not clarify whether they consider the feature extractor time in their detection time. Therefore, we presented the detection time results separately in Table 18 to evidence the time taken in each stage. The time taken in our feature extractor step was 60 μs, increasing the detection time by 56% and 27% for the attack detector stage in the AEID and TOW-IDS datasets, respectively. However, even with this increase, our results are still lower than the real-time detection threshold of 1,000 μs specified in (JEONG et al., 2021). Considering the time taken by the feature extractor step, our attack detector results are still the best for

the AEID dataset.

Although it is not possible to make a direct comparison between our results and the results presented in (HAN; KWAK; KIM, 2023), mainly due to their code implementation not being publicly available and the uncertainty regarding the platform the authors used to conduct the timing experiments, we can make a qualitative discussion of our results. As seen in Table 18, even with a worse computing platform (we have used an Intel Xeon CPU, while the authors of (HAN; KWAK; KIM, 2023) used a CPU and a 2080 RTX GPU), we have obtained a better result with our attack detector stage model. However, our detection results had a significant improvement decrease for the TOW-IDS specifically. This shows that if we investigate improvement approaches for the attack detector stage model and/or its feature extraction process, it could be a potential candidate for achieving similar detection results to the ones presented in (HAN; KWAK; KIM, 2023).

Table 18 – Detection time metrics for the compared works. The detection time of the feature extractor is 60 μs. *: works in which we reproduced the code to perform the timing experiments, **: works in which we have used the results presented by the authors.

Platform	**Method**	**Detection time (μs/sample)**	
		AEID dataset	**TOW-IDS dataset**
Intel(R) Xeon(R) CPU @ 2.20GHz	Our work (Attack detector)	**107**	**221**
	Our work (Attack classifier)	4510	4757
	(LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023)*	4510	-
	(CARMO et al., 2022)*	383	-
	(JEONG et al., 2021)*	7200	-
4790K CPU, 32GB RAM, and 2080 RTX GPU	(HAN; KWAK; KIM, 2023)**	-	250

Source: The author (2024)

We carried out an overall detection time analysis of our proposed IDS by applying Eq. (6.16) with the results presented in Tables 15, 16 and 18. Therefore, we obtained an $\mathrm{DT}_{\mathrm{overall}}$ of 116 μs per sample for the AEID dataset model and 774 μs per sample for the TOW-IDS dataset. These results demonstrate that our multi-stage approach has significantly improved the detection time criteria for automotive IDS and enhanced the possibility of using traditional machine learning alongside deep learning to obtain the best from both algorithms.

In Table 19, we present a summary of our IDS detection time containing the minimum, overall, and maximum detection times considering the influence of all IDS components: feature extractor, attack detector stage, and attack classifier stage. We consider the Equations (6.17), (6.18), and (6.19) to compute the timing summary metrics.

$$\text{Min} = \text{FE}_{\text{time}} + \text{DT}_{\text{attackdetector}} \tag{6.17}$$

$$\text{Overall} = \text{FE}_{\text{time}} + \text{DT}_{\text{overall}} \tag{6.18}$$

$$\text{Max} = \text{FE}_{\text{time}} + \text{DT}_{\text{attackclassifier}}, \tag{6.19}$$

where FE_{time} is the time taken by the feature extractor.

The summary detection time results presented in Table 19 demonstrate that even considering the feature extractor time, it has a minor impact on the overall timing performance of the proposed IDS. At last, our IDS timing criteria proposed in (6.1) were successfully satisfied and validated by our experimental results. We have obtained a $\text{DT}_{\text{attackdetector}}$ 42 times smaller than the $\text{DT}_{\text{attackclassifier}}$ for the AEID dataset models and 21 times smaller for the TOW-IDS dataset.

Table 19 – Minimum, overall, and maximum detection time summary considering the proposed IDS components.

Summary	**Detection time (μs)**	
	AEID	**TOW-IDS**
Min	167	281
Overall	176	834
Max	4570	4817

Source: The author (2024)

6.4.3 Limitations

While developing and evaluating our proposed IDS, we have identified limitations that could be further addressed. As it follows, we discuss these limitations to serve as a guide for future improvements.

Supervised training and zero-day attacks. Some of the state-of-the-art automotive Ethernet IDSs use a supervised training approach to learn the behaviors of both normal and known attack behaviors. Supervised training usually provided high detection accuracy to the detriment of not being able to detect attacks that were not in the training data, such as zero-day attacks. Our proposed IDS attack detector stage is trained to recognize both normal and general attack behaviors. In the case of a zero-day attack, which may exhibit a behavior that

differs from the norm, the IDS would still be able to identify it as an attack. One possible approach to improve zero-day attack detection is to use unsupervised or semi-supervised learning techniques to detect deviations from normal behavior instead of learning the normal and attack behavior. Another direction could rely on using probabilities instead of multi-class binary output and setting a threshold that could alert the possibility of a zero-day attack.

Feature engineering. Most state-of-the-art automotive Ethernet IDSs have a feature extractor that requires a domain specialist in its design and evaluation steps, which usually comes with a cost and may not be desirable or affordable. One possible approach to replace the feature extractor is to use additional layers in the DL architecture with this function. However, using a higher number of layers may increase the detection time of the IDS.

Adversarial attacks. DL-based IDSs are vulnerable to adversarial attacks, meaning an attacker can modify the input data to avoid detection and jeopardize the IDS. To address this problem, adversarial training can be considered, where the model is trained using clean and adversarially perturbed data. This will help to enhance the system's ability to detect and respond to potential threats.

7 CONCLUSIONS AND FUTURE WORK

In this chapter, we present a summarization of the results obtained in our previously presented contributions and highlight some possible future work that could be explored to advance the field of intrusion detection in automotive Ethernet networks.

7.1 SUMMARY OF RESULTS

In (LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023), we proposed a DL-based IDS for detecting replay attacks in an automotive Ethernet network. Unlike previous works, our IDS optimizes detection results, detection time, and storage size simultaneously during the training step by applying the LilNetX framework introduced in (GIRISH et al., 2022). The optimization process relies on adding two terms to the loss function: the storage size and the detection time. These terms update the network weights based on their transformations in latent space parameters, inducing a small bit representation and weights sparsity based on their entropy. The motivation behind optimizing detection time and storage size is that the target platform to deploy the proposed IDS is a microcontroller device, which is a resource-constrained device with memory limitations.

We have also analyzed the trade-off between detection results, detection time, and storage size. The balance of the abovementioned metrics is essential to designing and deploying an IDS in a resource-constrained environment such as an IVN. We have also compared our results with other state-of-the-art intrusion detection systems for automotive Ethernet networks. In (JEONG et al., 2021), the authors proposed a 2D-CNN IDS that achieved higher detection metrics but required GPU-based devices to achieve real-time detection. In (CARMO et al., 2022), the authors proposed a simpler model based on the XGBoost algorithm, which achieved good results regarding detection time in low-cost hardware but with the cost of a more significant drop in detection metrics.

Secondly, we proposed a novel multi-stage deep learning approach that consists of two stages. The attack detector stage aims to ensure a fast detection time, while the attack classifier stage focuses on achieving the most accurate results. We have used a Random Forest classifier in the attack detector stage and a Pruned CNN, as obtained in our previous work (LUZ; FREITAS DE ARAUJO-FILHO; CAMPELO, 2023), in the attack classifier stage.

We have evaluated our proposed IDS in two publicly available automotive Ethernet datasets: the AEID dataset (JEONG et al., 2021) and the TOW-IDS dataset (HAN; KWAK; KIM, 2023). We have also compared our experimental results with state-of-the-art works, and our attack classifier and attack detector stages obtained the best results among the works that used the AEID dataset, with F1-Score greater than 0.995. For the TOW-IDS dataset, our detection results only differed in the third decimal digit compared to IDS proposed in (HAN; KWAK; KIM, 2023). Furthermore, our proposed IDS presented a significant improvement in the detection time, obtaining an overall detection time result of 116 microseconds per sample for the AEID dataset and 774 microseconds per sample for the TOW-IDS dataset, being able to fulfill the real-time detection threshold of 1,000 μs/sample proposed in (JEONG et al., 2021).

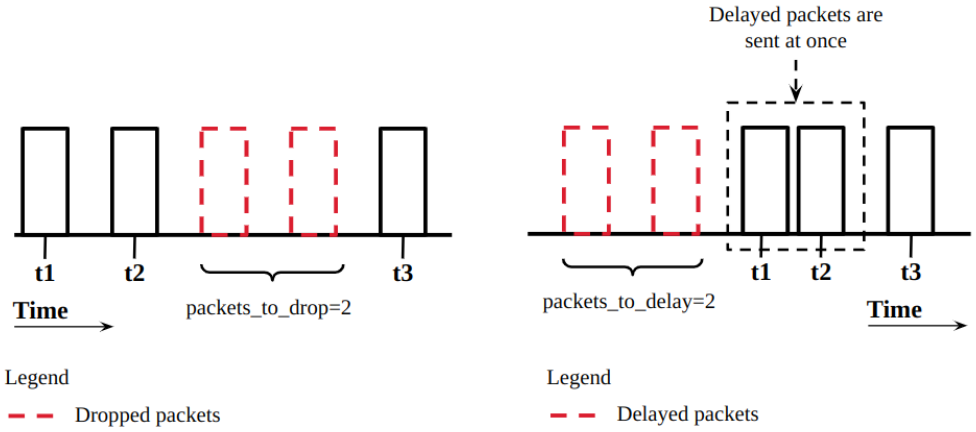
7.2 FUTURE WORK

We have identified several future work possibilities that could be employed to advance the field of IDSs for automotive Ethernet networks. As it follows, we highlight these possibilities with a brief description to provide an initial direction for other researchers:

Conducting cyberattacks with automotive-grade hardware. By the time this dissertation was written, only two datasets that cover cyberattacks in automotive Ethernet environments were publicly available: AEID and TOW-IDS datasets. Therefore, one of the author's future works is to conduct other cyberattacks, such as the frame drop (Fig. 37a) and frame delay and burst attacks (Fig 37b, which intends to manipulate how or if the end device receives the packets and propose new countermeasures to these attacks. These cyberattacks were conducted using automotive-grade devices of LIVE laboratory in CIn-UFPE (see Figure 38), which highlights the effort to expand the automotive Ethernet cybersecurity scenario within an essential step of improving the reliability of the proposed countermeasures by increasing the diversity of datasets for heterogeneous automotive networks. Besides the conduction of new cyberattacks, automotive-grade hardware can also be used to validate the feasibility of previously developed cyberattacks, such as the replay attack demonstrated in (JEONG et al., 2021).

Simulation environments. Another way to improve the reliability of the proposed countermeasures without the higher cost involved in purchasing automotive-grade devices or even real vehicles is to leverage the use of simulation tools to simulate a real-world environment for

Figure 37 – Other cyberattacks that can be conducted in automotive Ethernet networks.



(a) Frame drop attack effect on the network packets.

(b) Frame delay and burst attack effect on the network packets.

Source: The author (2024)

Figure 38 – Automotive-grade devices to simulate an automotive Ethernet network composed of an AVB Talker, a TAP and an AVB Listener to transmit audio signals.

Source: The author (2024)

vehicles where one can evaluate intrusion detection systems based in a simulated IVN. For instance, the CARLA simulator (DOSOVITSKIY et al., 2017), which is focused on autonomous driving research, could be extended to use a virtual CAN bus implemented with the SocketCAN (The Linux Kernel Organization, Inc., 2024) Linux facility, enabling the simulation of ECUs that uses the CAN protocol. Regarding automotive Ethernet networks, the OMNeT++ INET framework (INET Framework Developers, 2024) could be integrated with CARLA to consider automotive Ethernet protocols such as AVTP and gPTP. Once the simulation environment is properly set and configured, it can be further used to demonstrate and develop new cyberattacks, as well as evaluating IVN IDSs in more realistic scenarios than only the test set of datasets.

Lower level implementation of the proposed IDSs. Our proposed IDS design criteria focus on low detection time and low storage size to deploy the IDS within a low-cost device

such as a microcontroller. The obtained results demonstrated the feasibility of embedding such algorithms in memory-constrained devices based on the achieved storage size. In this direction, a possible evolution is to implement the proposed IDSs in a lower-level programming language such as C/C++ and interface the device with an AVB listener to evaluate the IDS in a testbed scenario more similar to in-vehicle network environments.

Feedback loop between stages. In our second contribution, we have proposed a multi-stage IDS that combines a fast detection time and accurate detection results in each stage. However, these stages are trained individually, and in more complex scenarios, the attack detector model may struggle to detect complex cyberattacks. In this direction, we plan to introduce a feedback loop from the attack classifier to the attack detector stage during the attack detector training phase. This feedback could improve the results by using the attack classifier stage during the training process of the attack detector stage. For instance, in cases where the attack detector stage makes a wrong detection and the attack classifier a successful detection, the wrong detection information could be used to improve the attack detector stage model parameters. In cases where the attack detector stage is a traditional ML algorithm, its parameters could be updated using online learning. On the other hand, if the attack detector stage turns out to be a DL algorithm, it opens up space to employ knowledge distillation techniques.

Explainable IDS. ML-based IDSs offer high detection accuracy in complex environments where it is difficult to create static rules. However, since most ML algorithms are black-box models, it is unclear how they make their decisions. We intend to use explainable models or frameworks like Trustee (proposed in (JACOBS et al., 2022a)) to overcome this issue. Incorporating explainability into the design of ML-based IDS can increase trust in how the algorithms make their decisions and assist forensics teams in tracing cyberattacks.

Unsupervised IDS. Additionally, as zero-day attacks are threats to supervised or signature-based IDSs, one potential direction is to develop an anomaly-based IDS using unsupervised ML algorithms. Traditional algorithms such as Isolation Forest and One-Class Support Vector Machines, Deep Learning Autoencoder architectures like Convolutional and Long-Short Term Memory Autoencoders (ALKHATIB et al., 2022), and even utilizing the Discriminator module of a Generative Adversarial Network (ARAUJO-FILHO et al., 2020).

Online training. Online training could be an alternative to periodic offline retraining through time to maintain the performance of our proposed IDSs. Additionally, Reinforcement Learning could be employed to update the IDS parameters once they are deployed automatically. However, this approach is more challenging because false positives can compromise the training

process.

IDS generalization. Using cross-validation and evaluating our proposed IDSs with held-out test sets improves the model generalization capability. However, these sets are still prone to sampling bias in the dataset creation. Therefore, there is room for improvement regarding IDS generalization capabilities for open-world environments where the data continuously changes, and contexts differ significantly from those in the training set. For instance, the authors of (MELO et al., 2022) proposed using federated learning with sampling methods and feature selection to improve the generalization capability of their proposed IDS in different networks.

Automotive Ethernet intrusion datasets and unbalanced data. As previously stated, the lack of a more representative amount of automotive Ethernet intrusion datasets is a limiting factor in developing new and more robust DL-based IDSs for automotive Ethernet networks. Besides that, the existing datasets suffer from data unbalancing. In this direction, to improve the ability of the IDSs to learn from malicious samples, oversampling techniques such as SMOTE proposed in (CHAWLA et al., 2002) and the more recent work of (LIN et al., 2022) that leverages the use of generative adversarial networks to generate samples from rare classes.

REFERENCES

ALKHATIB, N.; GHAUCH, H.; DANGER, J.-L. SOME/IP intrusion detection using deep learning-based sequential models in automotive ethernet networks. In: *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. [S.l.: s.n.], 2021. p. 0954–0962.

ALKHATIB, N.; MUSHTAQ, M.; GHAUCH, H.; DANGER, J.-L. Unsupervised network intrusion detection system for avtp in automotive ethernet networks. In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE Press, 2022. p. 1731–1738. Available at: <https://doi.org/10.1109/IV51971.2022.9827285>.

ARAUJO-FILHO, P. Freitas de; KADDOUM, G.; CAMPELO, D. R.; SANTOS, A. G.; MACÊDO, D.; ZANCHETTIN, C. Intrusion detection for cyber–physical systems using generative adversarial networks in fog environment. *IEEE Internet of Things Journal*, IEEE, v. 8, n. 8, p. 6247–6256, 2020.

BANDUR, V.; SELIM, G.; PANTELIC, V.; LAWFORD, M. Making the case for centralized automotive e/e architectures. *IEEE Transactions on Vehicular Technology*, IEEE, v. 70, n. 2, p. 1230–1245, 2021.

BELLO, L. L.; PATTI, G.; LEONARDI, L. A perspective on ethernet in automotive communications—current status and future trends. *Applied Sciences*, v. 13, n. 3, 2023. ISSN 2076-3417. Available at: <https://www.mdpi.com/2076-3417/13/3/1278>.

BIANCO, S.; CADENE, R.; CELONA, L.; NAPOLETANO, P. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, v. 6, p. 64270–64277, 2018.

BOSCH, R. *Bosch automotive electrics and automotive electronics: systems and components, networking and hybrid drive*. [S.l.]: Springer Vieweg., 2014.

BOSCH, R. et al. Can specification version 2.0. *Robert Bosch GmbH, Postfach*, v. 300240, p. 72, 1991.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, p. 5–32, 2001.

BURKE, K. *How Does a Self-Driving Car See?* 2019. <https://blogs.nvidia.com/blog/2019/04/15/how-does-a-self-driving-car-see/>. Accessed: 2022-12-30.

CAI, Z.; WANG, A.; ZHANG, W.; GRUFFKE, M.; SCHWEPPE, H. 0-days & mitigations: Roadways to exploit and secure connected bmw cars. *Black Hat USA*, v. 2019, n. 39, p. 6, 2019.

CARMO, P.; Freitas de Araujo-Filho, P.; CAMPELO, D.; FREITAS, E.; FILHO, A. O.; SADOK, D. Machine learning-based intrusion detection system for automotive ethernet: Detecting cyber-attacks with a low-cost platform. In: *Anais do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2022. p. 196–209. ISSN 2177-9384. Available at: <https://sol.sbc.org.br/index.php/sbrc/article/view/21171>.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, v. 16, p. 321–357, 2002.

CHECKOWAY, S.; MCCOY, D.; KANTOR, B.; ANDERSON, D.; SHACHAM, H.; SAVAGE, S.; KOSCHER, K.; CZESKIS, A.; ROESNER, F.; KOHNO, T. Comprehensive experimental analyses of automotive attack surfaces. In: *20th USENIX Security Symposium (USENIX Security 11)*. San Francisco, CA: USENIX Association, 2011. Available at: <https://www.usenix.org/conference/usenix-security-11/comprehensive-experimental-analyses-automotive-attack-surfaces>.

CHO, K.-T.; SHIN, K. G. Fingerprinting electronic control units for vehicle intrusion detection. In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016. p. 911–927. ISBN 978-1-931971-32-4. Available at: <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cho>.

CHO, K.-T.; SHIN, K. G. Viden: Attacker identification on in-vehicle networks. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2017. (CCS '17), p. 1109–1123. ISBN 9781450349468. Available at: <https://doi.org/10.1145/3133956.3134001>.

CHOI, W.; JOO, K.; JO, H. J.; PARK, M. C.; LEE, D. H. Voltageids: Low-level communication characteristics for automotive intrusion detection system. *IEEE Transactions on Information Forensics and Security*, v. 13, n. 8, p. 2114–2129, 2018.

CORREA, C.; KOZIEROK, C.; SIMON, J.; GUBOW, M.; BHAGWAT, S. *Automotive Ethernet: The Definitive Guide*. [S.l.]: Intrepid Control Systems, 2014. ISBN 9780990538820.

CyCraft Athena. *Automotive ATT&CK Matrix*. 2024. Accessed: 2024-08-10. Available at: <https://athena.cycraft.ai/matrices/automotive/>.

DAŞ, R.; KARABADE, A.; TUNA, G. Common network attack types and defense mechanisms. In: IEEE. *2015 23nd Signal Processing and Communications Applications Conference (SIU)*. [S.l.], 2015. p. 2658–2661.

DOSOVITSKIY, A.; ROS, G.; CODEVILLA, F.; LOPEZ, A.; KOLTUN, V. CARLA: An open urban driving simulator. In: *Proceedings of the 1st Annual Conference on Robot Learning*. [S.l.: s.n.], 2017. p. 1–16.

FOTOUHI, M.; BUSCEMI, A.; BOUALOUACHE, A.; JOMRICH, F.; KOEBEL, C.; ENGEL, T. Assessing the impact of attacks on an automotive ethernet time synchronization testbed. In: *2023 IEEE Vehicular Networking Conference (VNC)*. [S.l.: s.n.], 2023. p. 223–230.

FRANKLE, J.; CARBIN, M. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018. Available at: <http://arxiv.org/abs/1803.03635>.

Freitas De Araujo-Filho, P.; PINHEIRO, A. J.; KADDOUM, G.; CAMPELO, D. R.; SOARES, F. L. An Efficient Intrusion Prevention System for CAN: Hindering Cyber-attacks with a Low-cost Platform. *IEEE Access*, p. 1–1, 2021.

GHOSAL, A.; CONTI, M. Security issues and challenges in V2X : A Survey. *Computer Networks*, Elsevier B.V., v. 169, p. 107093, 2020. ISSN 1389-1286. Available at: <https://doi.org/10.1016/j.comnet.2019.107093>.

GIRISH, S.; GUPTA, K.; SINGH, S.; SHRIVASTAVA, A. *LilNetX: Lightweight Networks with EXtreme Model Compression and Structured Sparsification*. arXiv, 2022. Available at: <https://arxiv.org/abs/2204.02965>.

GMIDEN, M.; GMIDEN, M. H.; TRABELSI, H. An intrusion detection method for securing in-vehicle can bus. In: IEEE. *2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. [S.l.], 2016. p. 176–180.

HAN, M. L.; KWAK, B. I.; KIM, H. K. TOW-IDS: Intrusion detection system based on three overlapped wavelets for automotive ethernet. *IEEE Transactions on Information Forensics and Security*, v. 18, p. 411–422, 2023.

HARTWICH, F.; BOSCH, R. *Introducing can xl into can networks*. 2020.

HARTWICH, F. et al. Can with flexible data-rate. In: CITESEER. *Proc. iCC*. [S.l.], 2012. p. 1–9.

IEEE. IEEE standard for layer 2 transport protocol for time sensitive applications in a bridged local area network. *IEEE Std 1722-2011*, p. 1–65, 2011.

IEEE. IEEE standard for a transport protocol for time-sensitive applications in bridged local area networks. *IEEE Std 1722-2016 (Revision of IEEE Std 1722-2011)*, p. 1–233, 2016.

IEEE. IEEE standard for a transport protocol for time-sensitive applications in bridged local area networks. *IEEE Std 1722-2016 (Revision of IEEE Std 1722-2011)*, p. 1–233, 2016.

IEEE. IEEE standard for local and metropolitan area networks-media access control (mac) security. *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)*, p. 1–239, 2018.

IEEE. IEEE standard for local and metropolitan area networks–timing and synchronization for time-sensitive applications. *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, p. 1–421, 2020.

INET Framework Developers. *INET Framework for OMNeT++*. 2024. Accessed: 2024-10-08. Available at: <https://inet.omnetpp.org/>.

JACOBS, A. S.; BELTIUKOV, R.; WILLINGER, W.; FERREIRA, R. A.; GUPTA, A.; GRANVILLE, L. Z. Ai/ml and network security: The emperor has no clothes. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2022. (CCS '22).

JACOBS, A. S.; BELTIUKOV, R.; WILLINGER, W.; FERREIRA, R. A.; GUPTA, A.; GRANVILLE, L. Z. Ai/ml for network security: The emperor has no clothes. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2022. (CCS '22), p. 1537–1551. ISBN 9781450394505. Available at: <https://doi.org/10.1145/3548606.3560609>.

JEONG, S.; JEON, B.; CHUNG, B.; KIM, H. K. Convolutional neural network-based intrusion detection system for AVTP streams in automotive ethernet-based networks. *Vehicular Communications*, v. 29, p. 100338, 2021. ISSN 2214-2096. Available at: <https://www.sciencedirect.com/science/article/pii/S2214209621000073>.

JEONG, S.; KIM, H. K.; HAN, M. L.; KWAK, B. I. AERO: Automotive ethernet real-time observer for anomaly detection in in-vehicle networks. *IEEE Transactions on Industrial Informatics*, p. 1–12, 2023.

JO, H. J.; CHOI, W. A Survey of Attacks on Controller Area Networks and Corresponding Countermeasures. *IEEE Transactions on Intelligent Transportation Systems*, p. 1–19, 2021. ISSN 15580016.

KOSCHER, K.; CZESKIS, A.; ROESNER, F.; PATEL, S.; KOHNO, T.; CHECKOWAY, S.; MCCOY, D.; KANTOR, B.; ANDERSON, D.; SHACHAM, H. et al. Experimental security analysis of a modern automobile. In: IEEE. *2010 IEEE symposium on security and privacy*. [S.l.], 2010. p. 447–462.

LAI, C.; LU, R.; ZHENG, D.; SHEN, X. Security and privacy challenges in 5g-enabled vehicular networks. *IEEE Network*, v. 34, n. 2, p. 37–45, 2020.

LAMPE, B.; MENG, W. Ids for can: A practical intrusion detection system for can bus security. In: IEEE. *GLOBECOM 2022-2022 IEEE Global Communications Conference*. [S.l.], 2022. p. 1782–1787.

LAMPE, B.; MENG, W. *can-train-and-test: A Curated CAN Dataset for Automotive Intrusion Detection*. 2023.

LAUSER, T.; ZELLE, D.; KERN, D.; KRAUSS, C.; VÖLKER, L. Security protocols for ethernet-based in-vehicle communication. In: IEEE. *2024 IEEE Vehicular Networking Conference (VNC)*. [S.l.], 2024. p. 148–155.

LEE, H.; JEONG, S. H.; KIM, H. K. Otids: A novel intrusion detection system for in-vehicle network by using remote frame. In: IEEE. *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. [S.l.], 2017. p. 57–5709.

LIN, Z.; LIANG, H.; FANTI, G.; SEKAR, V. Raregan: Generating samples for rare classes. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2022. v. 36, n. 7, p. 7506–7515.

LIU, J.; ZHANG, S.; SUN, W.; SHI, Y. In-vehicle network attacks and countermeasures: Challenges and future directions. *IEEE Network*, v. 31, n. 5, p. 50–58, 2017. ISSN 08908044.

LOCAL, I. S. for; NETWORKS-BRIDGES, M. A.; NETWORKS, B. *Amendment 28: Per-Stream Filtering and Policing*. [S.l.]: IEEE Piscataway, NJ. USA, 2017.

LUZ, L.; FREITAS DE ARAUJO-FILHO, P.; CAMPELO, D. Multi-criteria optimized deep learning-based intrusion detection system for detecting cyberattacks in automotive ethernet networks. In: *Anais do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2023. p. 197–210. ISSN 2177-9384. Available at: <https://sol.sbc.org.br/index.php/sbrc/article/view/24539>.

LUZ, L. F. M. da; FREITAS DE ARAUJO-FILHO, P.; CAMPELO, D. R. Multi-stage deep learning-based intrusion detection system for automotive ethernet networks. *Ad Hoc Networks*, Elsevier, v. 162, p. 103548, 2024.

MARCHETTI, M.; STABILI, D. Anomaly detection of can bus messages through analysis of id sequences. In: IEEE. *2017 IEEE intelligent vehicles symposium (IV)*. [S.l.], 2017. p. 1577–1583.

MATHEUS, K.; KÖNIGSEDER, T. *Automotive Ethernet*. [S.l.]: Cambridge University Press, 2021.

MELO, L. H. de; BERTOLI, G. de C.; PEREIRA, L. A.; SAOTOME, O.; DOMINGUES, M. F.; SANTOS, A. L. dos. Generalizing flow classification for distributed denial-of-service over different networks. In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. [S.l.: s.n.], 2022. p. 879–884.

MILLER, C.; VALASEK, C. Remote Exploitation of an Unaltered Passenger Vehicle. *Defcon 23*, v. 2015, p. 1–91, 2015. Available at: <http://illmatics.com/RemoteCarHacking.pdf>.

MITRE ATT&CK. *MITRE ATT&CK Framework*. 2024. Accessed: 2024-08-10. Available at: <https://attack.mitre.org/>.

MOUSSA, B.; KASSOUF, M.; HADJIDJ, R.; DEBBABI, M.; ASSI, C. An extension to the precision time protocol (PTP) to enable the detection of cyber attacks. *IEEE Transactions on Industrial Informatics*, IEEE, v. 16, n. 1, p. 18–27, 2019.

NIE, S.; LIU, L.; DU, Y. Free-fall: Hacking tesla from wireless to can bus. *Briefing, Black Hat USA*, v. 25, n. 1, p. 16, 2017.

NIE, S.; LIU, L.; DU, Y.; ZHANG, W. Over-the-air: How we remotely compromised the gateway, bcm, and autopilot ecus of tesla cars. *Briefing, Black Hat USA*, v. 91, p. 1–19, 2018.

NISIOTI, A.; MYLONAS, A.; YOO, P. D.; KATOS, V. From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. *IEEE Communications Surveys & Tutorials*, IEEE, v. 20, n. 4, p. 3369–3388, 2018.

OKTAY, D.; BALLÉ, J.; SINGH, S.; SHRIVASTAVA, A. Scalable model compression by entropy penalized reparameterization. *arXiv preprint arXiv:1906.06624*, 2019.

PARET, D.; REBAINE, H. *Autonomous and Connected Vehicles: Network Architectures from Legacy Networks to Automotive Ethernet*. [S.l.]: John Wiley & Sons, 2022.

QUADAR, N.; CHEHRI, A.; DEBAQUE, B.; AHMED, I.; JEON, G. Intrusion detection systems in automotive ethernet networks: Challenges, opportunities and future research trends. *IEEE Internet of Things Magazine*, IEEE, v. 7, n. 2, p. 62–68, 2024.

RESCORLA, E. *The transport layer security (TLS) protocol version 1.3*. [S.l.], 2018.

SEO, E.; SONG, H. M.; KIM, H. K. GIDS: GAN based intrusion detection system for in-vehicle network. In: *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. [S.l.: s.n.], 2018. p. 1–6.

SHIBLY, K. H.; HOSSAIN, M. D.; INOUE, H.; TAENAKA, Y.; KADOBAYASHI, Y. A feature-aware semi-supervised learning approach for automotive ethernet. In: *2023 IEEE International Conference on Cyber Security and Resilience (CSR)*. [S.l.: s.n.], 2023. p. 426–431.

SONG, H. M.; KIM, H. R.; KIM, H. K. Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network. In: IEEE. *2016 international conference on information networking (ICOIN)*. [S.l.], 2016. p. 63–68.

SONG, H. M.; WOO, J.; KIM, H. K. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, Elsevier, v. 21, p. 100198, 2020.

SPARTA. *Space Attack Research & Tactic Analysis*. 2024. Accessed: 2024-08-10. Available at: <https://sparta.aerospace.org/>.

STANDARDIZATION, I. O. for. *ISO/SAE 21434: 2021: Road Vehicles: Cybersecurity Engineering*. [S.l.]: ISO, 2021.

STROM, B. *Getting Started with ATT&CK - Adversary Emulation and Red Teaming*. 2020. Accessed: 2024-08-10. Available at: <https://medium.com/mitre-attack/getting-started-with-attack-red-29f074ccf7e3>.

SUN, X.; YU, F. R.; ZHANG, P. A survey on cyber-security of connected and autonomous vehicles (CAVs). *IEEE Transactions on Intelligent Transportation Systems*, v. 23, n. 7, p. 6240–6259, 2022.

The Linux Kernel Organization, Inc. *Controller Area Network (CAN) Subsystem for Linux*. 2024. Accessed: 2024-10-08. Available at: <https://docs.kernel.org/networking/can.html>.

TINDELL, K. *CAN Injection: keyless car theft*. 2023. Accessed: 2024-07-25. Available at: <https://kentindell.github.io/2023/04/03/can-injection/>.

TUOHY, S.; GLAVIN, M.; HUGHES, C.; JONES, E.; TRIVEDI, M.; KILMARTIN, L. Intra-Vehicle Networks: A Review. *IEEE Transactions on Intelligent Transportation Systems*, v. 16, n. 2, p. 534–545, 2015. ISSN 15249050.

UN Regulation 155. *UN Regulation No. 155 - Cyber security and cyber security management system*. 2021. <https://unece.org/transport/documents/2021/03/standards/un-regulation-no-155-cyber-security-and-cyber-security>.

VAL, I.; SEIJO, O.; TORREGO, R.; ASTARLOA, A. IEEE 802.1AS clock synchronization performance evaluation of an integrated wired–wireless tsn architecture. *IEEE Transactions on Industrial Informatics*, v. 18, n. 5, p. 2986–2999, 2022.

VERMA, M. E.; IANNACONE, M. D.; BRIDGES, R. A.; HOLLIFIELD, S. C.; MORIANO, P.; KAY, B.; COMBS, F. L. *Addressing the Lack of Comparability & Testing in CAN Intrusion Detection Research: A Comprehensive Guide to CAN IDS Data & Introduction of the ROAD Dataset*. 2022.

VicOne. *Mapping Automotive Threats to Perform Threat Investigations*. 2024. Accessed: 2024-08-10. Available at: <https://vicone.com/files/use_case_mapping-automotive-threats-to-perform-threat-investigations.pdf>.

VINCENZI, M. D.; COSTANTINO, G.; MATTEUCCI, I.; FENZL, F.; PLAPPERT, C.; RIEKE, R.; ZELLE, D. A systematic review on security attacks and countermeasures in automotive ethernet. *ACM Computing Surveys*, ACM New York, NY, v. 56, n. 6, p. 1–38, 2024.

WANG, Q.; LU, Z.; QU, G. An entropy analysis based intrusion detection system for controller area network in vehicles. In: IEEE. *2018 31st IEEE International System-on-Chip Conference (SOCC)*. [S.l.], 2018. p. 90–95.

WU, W.; HUANG, Y.; KURACHI, R.; ZENG, G.; XIE, G.; LI, R.; LI, K. Sliding window optimized information entropy analysis method for intrusion detection on in-vehicle networks. *Ieee Access*, IEEE, v. 6, p. 45233–45245, 2018.

WU, W.; LI, R.; XIE, G.; AN, J.; BAI, Y.; ZHOU, J.; LI, K. A survey of intrusion detection for in-vehicle networks. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 21, n. 3, p. 919–933, 2020. ISSN 15580016.

WUNDER, J. *Getting Started with ATT&CK: Detection and Analytics*. 2020. Accessed: 2024-08-10. Available at: <https://medium.com/mitre-attack/getting-started-with-attack-detection-a8e49e4960d0>.

YANG, L.; MOUBAYED, A.; HAMIEH, I.; SHAMI, A. Tree-based intelligent intrusion detection system in internet of vehicles. In: *2019 IEEE Global Communications Conference (GLOBECOM)*. [S.l.: s.n.], 2019. p. 1–6.