Multi-Stage Deep Learning-based Intrusion Detection System for Automotive Ethernet Networks

Luigi F. Marques da Luz[a,b,*], Paulo Freitas de Araujo-Filho[a] and Divanilson R. Campelo[a]

[a]Centro de Informática – Universidade Federal de Pernambuco (CIn - UFPE), Av. Jorn. Aníbal Fernandes – s/n, Recife, 50740-560, PE, Brazil
[b]Centro de Estudos e Sistemas Avançados do Recife (CESAR), Rua Bione - 220, Recife, 50030-390, PE, Brazil

ARTICLE INFO

Keywords:
Intrusion Detection System
Multi-stage
Deep Learning
Automotive Ethernet

ABSTRACT

Modern automobiles are increasing the demand for automotive Ethernet as a high-bandwidth and flexible in-vehicle network technology. However, since Ethernet does not have native support for authentication or encryption, intrusion detection systems (IDSs) are becoming an attractive security mechanism to detect malicious activities that may affect Ethernet-based communication in cars. This paper proposes 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.

1. Introduction

Today's cars contain dozens of electronic control units (ECUs), which are interconnected by a number of in-vehicle networks (IVNs), Lai, Lu, Zheng and Shen (2020a); Wu, Li, Xie, An, Bai, Zhou and Li (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. 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 and 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 and Königseder (2021); Tuohy, Glavin, Hughes, Jones, Trivedi and Kilmartin (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, Zhang, Sun and Shi (2017); Jo and Choi (2021); Ghosal and Conti (2020). Checkoway, McCoy, Kantor, Anderson, Shacham, Savage, Koscher, Czeskis, Roesner and Kohno (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 and 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, Jeon, Chung and Kim (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 and 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

*Corresponding author
lfml@cin.ufpe.br (L.F.M.d. Luz); pfreitas@cin.ufpe.br (P.F.d. Araujo-Filho); dcampelo@cin.ufpe.br (D.R. Campelo)
ORCID(s): 0009-0000-8781-9219 (L.F.M.d. Luz)

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, Mylonas, Yoo and Katos (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 to identify normal profiles, machine learning (ML) based IDSs have gained attention because of their detection results and capability of detecting more complex attacks Freitas de Araujo-Filho, Kaddoum, Campelo, Santos, Macêdo and Zanchettin (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, Lu, Zheng and Shen (2020b); Wu et al. (2020). However, ML-based IDSs usually demand high computational power, often unavailable in IVNs Bianco, Cadene, Celona and Napoletano (2018). Therefore, the open challenges regarding the development of IVN IDSs are related to 1) a low detection time to prevent cyberattacks before causing damage and 2) classify the malicious activity to provide information for forensics and future improvements Wu et al. (2020). 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.

Contributions. This paper proposes a novel multi-stage deep learning-based IDS. The first stage has the goal of quickly detecting cyberattacks so that it is possible to stop attacks before they can cause damage. The second stage, on the other hand, is responsible for achieving a detection with a lower false positive rate when compared to the first, as well as the cyberattack classification. We aim to distinguish and classify benign and malicious in-vehicle network protocol packets in a heterogeneous automotive network. The considered in-vehicle network contains legitimate and malicious packets from AVTP, gPTP, and CAN protocols, which cover typical applications of video stream transmission, synchronization, and legacy protocol for safety-critical systems. In a nutshell, the main contributions of this work are:

- We propose a novel multi-stage deep learning-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;
- We evaluate our proposed IDS in publicly available automotive Ethernet datasets and compare our experimental results with state-of-the-art automotive Ethernet IDSs regarding their detection metrics and detection time.

This paper is organized as follows. In Section 2, we present the recent advances in automotive Ethernet IDSs. Section 3 presents the threat model considered for our work. Section 4 introduces the architecture of our proposed IDS and describes the modeling criteria and each selected method in the multi-stage approach. Section 5 explains the experimental setup and methodology used to evaluate our IDS. In Section 6, we show the results of the IDS and compare them to those of state-of-the-art automotive Ethernet IDSs in the literature. Finally, Section 7 concludes our paper and discusses future works.

2. Related Work

Recently, intrusion detection systems for automotive Ethernet networks have been proposed in the literature. The authors in Jeong et al. (2021) proposed an IDS that uses a 2D-convolutional neural network (2D-CNN) to detect replay attacks in AVTP packets. The authors also released their dataset of injection attacks for public use. However, their IDS cannot detect unknown attacks: the IDS is based on supervised learning, requiring labeled data, which can be challenging to obtain. Additionally, the model needs GPUs to achieve short detection times, leading to high deployment costs.

In Alkhatib, Ghauch and Danger (2021), the authors evaluated the performance of deep learning (DL)-based IDSs for detecting cyberattacks in automotive Ethernet. However, they focused only on using DL techniques to detect attacks in an offline intrusion detection scenario, i.e., it is impossible to prevent the cyberattack as it happens only in a posteriori-analysis step. Another downside is that their method is made specifically for scalable service-oriented middleware over IP (SOME/IP) networks and cannot be used with other protocols.

In Carmo, Freitas de Araujo-Filho, Campelo, Freitas, Filho and Sadok (2022), the authors suggested using the XGBoost algorithm to identify replay attacks in AVTP packets. Their proposal detected attacks in 620 µs/sample using low-cost, CPU-based hardware such as Raspberry Pi. However, one should note that the XGBoost model is designed for tabular data analysis and is unsuitable for detecting spatial or time relationships in network traffic images or packets.

In Alkhatib, Mushtaq, Ghauch and Danger (2022), the authors evaluated the detection time, model size, and detection-related metrics of two autoencoder-based models, a convolutional autoencoder (CAE) and a long short-term memory-based autoencoder (LSTM-AE) to develop an anomaly detector for detecting zero-day cyberattacks in AVTP packets. Despite the ability to detect new attacks, their proposed IDSs had a slight decrease in detection performance when

Table 1
Related work. AEID and TOW-IDS refer to the datasets proposed in Jeong et al. (2021) and Han et al. (2023), respectively. SOME/IP refers to the dataset used in Alkhatib et al. (2021)

Reference	Method	Dataset	Supervised	Multi-label	Timing requirements
Jeong et al. (2021)	2D-CNN	AEID	Yes	No	Yes, with GPU devices
Alkhatib et al. (2021)	DL methods	SOME/IP	Yes	No	No, offline IDS
Carmo et al. (2022)	XGBoost	AEID	Yes	No	Yes
Luz et al. (2023)	Pruned and quantized 2DCNN	AEID	Yes	No	Yes
Alkhatib et al. (2022)	CAE and LSTMAE	AEID	No	No	No
Han et al. (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
Our work	**Multi-stage IDS**	**AEID and TOW-IDS**	**Yes**	**Yes**	**Yes**

compared to other state-of-the-art works Jeong et al. (2021); Carmo et al. (2022).

In Luz, Freitas de Araujo-Filho and Campelo (2023), we proposed 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.

As it follows, the authors of Han, Kwak and Kim (2023) presented a dataset of a heterogeneous automotive Ethernet network containing packets from the protocols AVTP, CAN over user datagram protocol (UDP), and gPTP. Their dataset comprised several new cyberattacks, such as CAM table overflow and PTP synchronization attack. The authors have also proposed a new feature extraction method using wavelet transforms and an IDS based on a Deep Convolutional Neural Network Model. 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.

The work presented in Shibly, Hossain, Inoue, Taenaka and Kadobayashi (2023) proposes a technique that demands less labeled data to train their proposed IDS by employing Generative Adversarial Networks (GAN). By doing so, they decrease the need for massive labeled datasets to train ML/DL-based IDSs for automotive networks. However, the presented results still need further improvement to match the detection performance of other methods, such as the one proposed in Han et al. (2023).

At last, the authors of Jeong, Kim, Han and Kwak (2023) have proposed a multimodal feature extractor that can gather information regarding protocol change, payload, and packet timestamps. They have also proposed an unsupervised deep neural network architecture to detect cyberattacks. Despite the ability to detect novel attacks, it cannot classify the kind of attack and demands to be deployed in GPU devices to achieve real-time detection.

Table 1 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.

Our proposed technique stands out from the existing works in detecting cyberattacks quickly and efficiently. It uses a divide-and-conquer strategy to identify suspicious events and then accurately classifies them among the known cyberattacks. This information can be communicated to the user and other systems in real-time to prevent further damage.

3. Threat model

In the threat model, we assume an attacker can access the in-vehicle network and send malicious frames through compromised AVTP talker and CAN nodes. This is illustrated in Fig. 1, where we also present an in-vehicle network 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 in-vehicle networks (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 2. The following

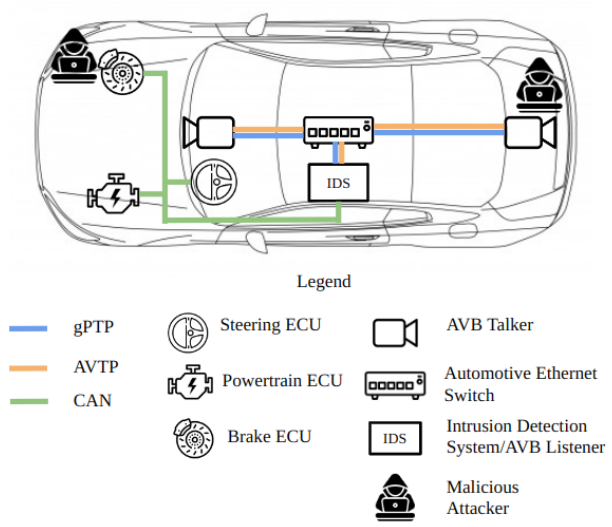


Figure 1: In-vehicle network architecture depicting its protocols, components, and compromised nodes.

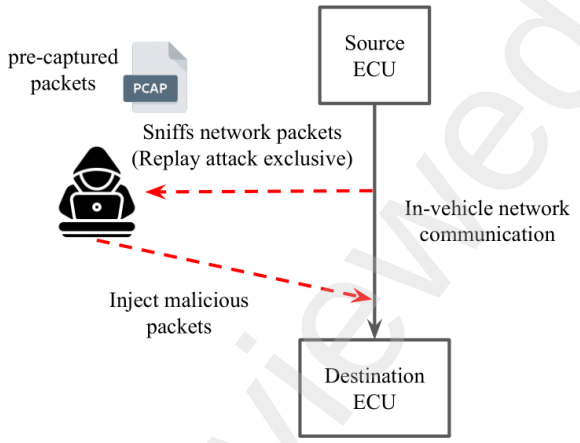


Figure 2: Replay and injection attacks for an arbitrary in-vehicle network.

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. 2. 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 and Valasek (2015), where the attackers injected a stop engine command with the vehicle in a highway.

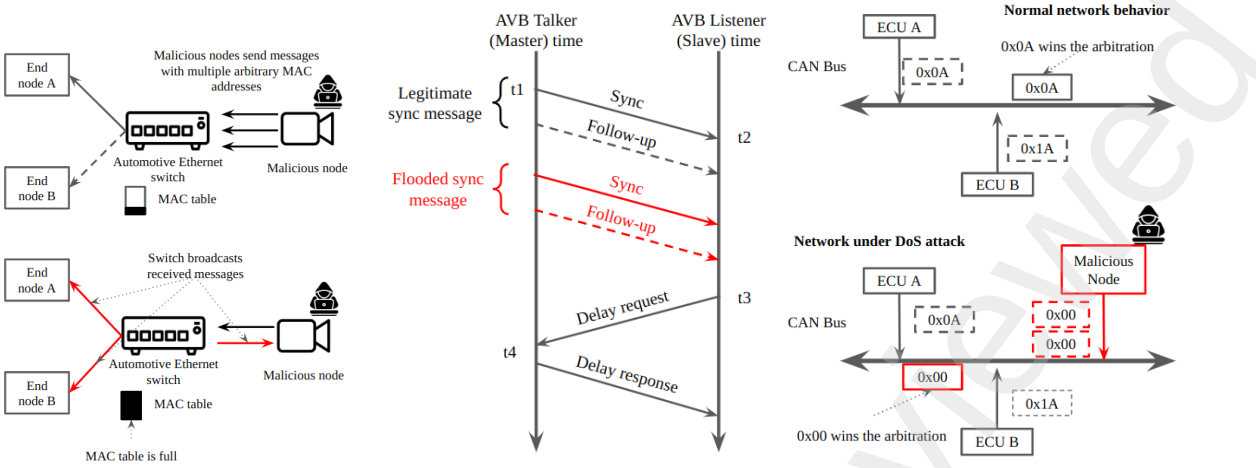
Replay attack. In Fig. 2, we present the working principle of the replay attack for an arbitrary IVN. It is worth mentioning that the capture and injection process depends on the IVN network topology and technologies. The replay attack is a specific injection attack where pre-captured frames or packets 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, Pinheiro, Kaddoum, Campelo and Soares (2021), but Han et al. (2023) were the first to bring CAN packets into an automotive Ethernet network by encapsulating it in UDP packets.

MAC flooding attack. Fig. 3a 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 Media Access Control (MAC) table, which stores known MAC addresses common to the network Daş, Karabade and 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 et al. (2023).

PTP sync attack. The gPTP protocol synchronizes the clocks of master and slave 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 master node Val, Seijo, Torrego and Astarloa (2022). The PTP sync attack is presented in Fig. 3b 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, Kassouf, Hadjidj, Debbabi and Assi (2019).

CAN denial of service attack. In the CAN bus arbitration, messages with lower IDs have higher priority Jo and Choi (2021). This concept is crucial to understanding the CAN denial of service (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 and Meng (2023); Verma, Iannacone, Bridges, Hollifield, Moriano, Kay and Combs (2022); Song, Woo and Kim (2020); Seo, Song and Kim (2018). The CAN replay attack was brought to an automotive Ethernet network by Han et al. (2023) by tunneling the CAN frames in UDP. Despite the significant impact of DoS in the in-vehicle network, it can be easily detected by knowing the valid CAN IDs and blocking messages from unknown IDs. In Fig. 3c, we visually



(a) MAC flooding attack. (b) PTP sync attack. (c) CAN DoS attack.

Figure 3: Block diagram for some of the considered in-vehicle network cyberattacks for this work.

Table 2
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

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 2 a summary of the previously described cyberattacks, the in-vehicle network protocols they aim to compromise, and their impacts.

4. Proposed IDS Architecture

This section describes our proposed IDS architecture, including its modeling criteria and the specification of its main components. Fig. 4 presents a block diagram of our proposed IDS major components: the feature extractor, first, and second stages.

Our proposed detection process is presented in Algorithm 1. 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. 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 first stage quickly determines if the received packets are normal or malicious and raises an alert if it finds any malicious packets. The second stage then determines the packet group's final classification and identifies the attack type if the data is found to be malicious.

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 and Zhang (2022); Jo and Choi (2021); Wu et al. (2020). The first stage prioritizes a fast detection of potential cyberattacks, as shown in Eq. (1):

$$\mathrm{DT}_{\mathrm{firststage}} << \mathrm{DT}_{\mathrm{secondstage}}, \tag{1}$$

where $\mathrm{DT}_{\mathrm{firststage}}$ and $\mathrm{DT}_{\mathrm{secondstage}}$ are the detection times of the first and second stages, respectively.

According to the abovementioned requirements, the desirable characteristics of the first-stage models 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

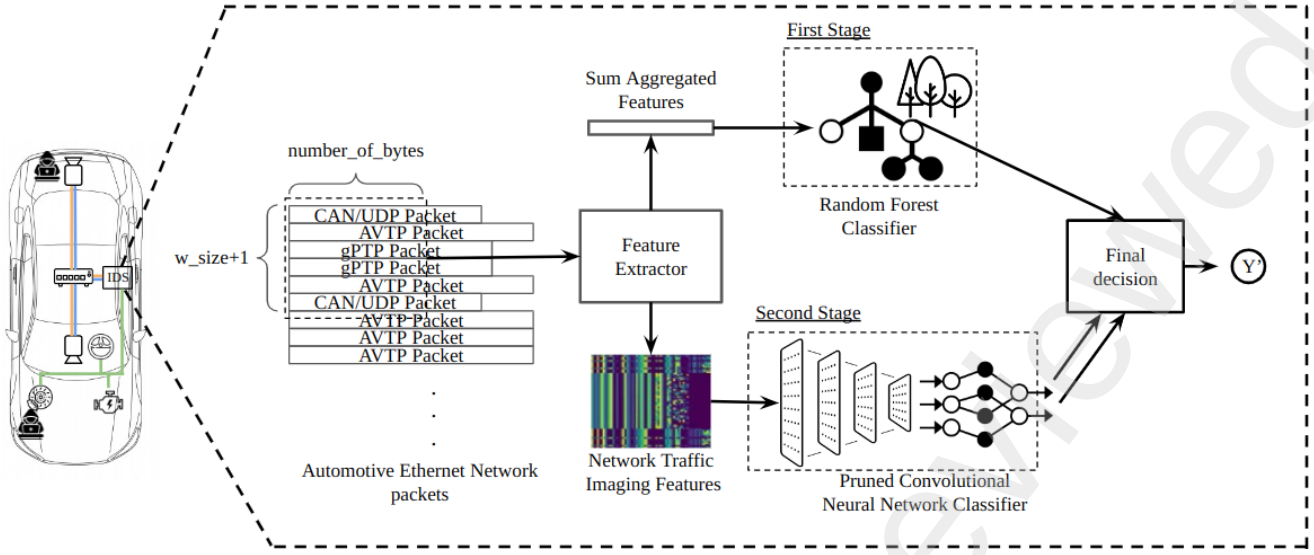


Figure 4: Block diagram of our proposed IDS main components.

Algorithm 1 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 first stage
y_2: Multi-class output of IDS second stage
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($\boldsymbol{x}_{raw}$)
    y_1 <- FirstStage($\boldsymbol{x_1}$)
    y_2 <- SecondStage($\boldsymbol{X_2}$)
    while SecondStage 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

automotive networks Carmo et al. (2022); Yang, Moubayed, Hamieh and Shami (2019); Freitas De Araujo-Filho et al. (2021).

On the other hand, the second stage is primarily responsible for detecting cyberattacks with higher detection metrics than the first stage, as represented in Eq. (2):

$$\mathrm{DM}_{\mathrm{secondstage}} > \mathrm{DM}_{\mathrm{firststage}} \tag{2}$$

where $\mathrm{DM}_{\mathrm{secondstage}}$ and $\mathrm{DM}_{\mathrm{firststage}}$ represent the overall detection metrics of the second and first stages, respectively. The detection metrics will be further described in 5.3.

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 et al. (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.

4.1. Feature extractor

Our proposed feature extractor is an expanded version of the feature generator initially proposed in Luz et al. (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. (3):

$$\boldsymbol{x}_{raw} = \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{\mathtt{w\_size}} \end{bmatrix}, \tag{3}$$

where each $\boldsymbol{x}_i$, $0 \leq i \leq \mathtt{w\_size}$, 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 × `n_bytes`, and contains the selected amount of `n_bytes` from the raw network packets:

$$\boldsymbol{X_{padded}} = \begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0\texttt{n\_bytes}} \\ a_{10} & a_{11} & \cdots & a_{1\texttt{n\_bytes}} \\ \vdots & \vdots & \ddots & \vdots \\ a_{\texttt{w\_size}0} & a_{\texttt{w\_size}1} & \cdots & a_{\texttt{w\_sizen\_bytes}} \end{bmatrix} \quad (4)$$

whose elements $a_{ij}$are defined by Eq. (5):

$$a_{ij} = \begin{cases} x_{ij}, & \text{if } 0 \leq j \leq \texttt{packet\_bytes} \\ 0, & \text{if } \texttt{packet\_bytes} < j \leq \texttt{n\_bytes} \end{cases} \quad (5)$$

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)

$$b_{ij} = \mod\left(\frac{a_{i+1j} - a_{ij}}{255}\right), \quad (6)$$

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. (7)

$$n_{ij} = \begin{cases} b_{i((j+1)/2)} >> 4, & \text{if } j \text{ is odd} \\ b_{i((j+1)/2)} \text{bitwise and 0x0F}, & \text{if } j \text{ is even} \end{cases}, \quad (7)$$

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 second stage 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 first-stage 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. (8)

$$c_i = \sum_{j=0}^{\texttt{w\_size-1}} b_{ij}. \quad (8)$$

In Fig. 5, 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 et al. (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 et al. (2023) is 9 bytes, and the CAN payload is stored before the 58th byte of the UDP frame.

4.2. First 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 first stage 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 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 3, 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 as long as they are increased, the weak learners become more prone to overfit.

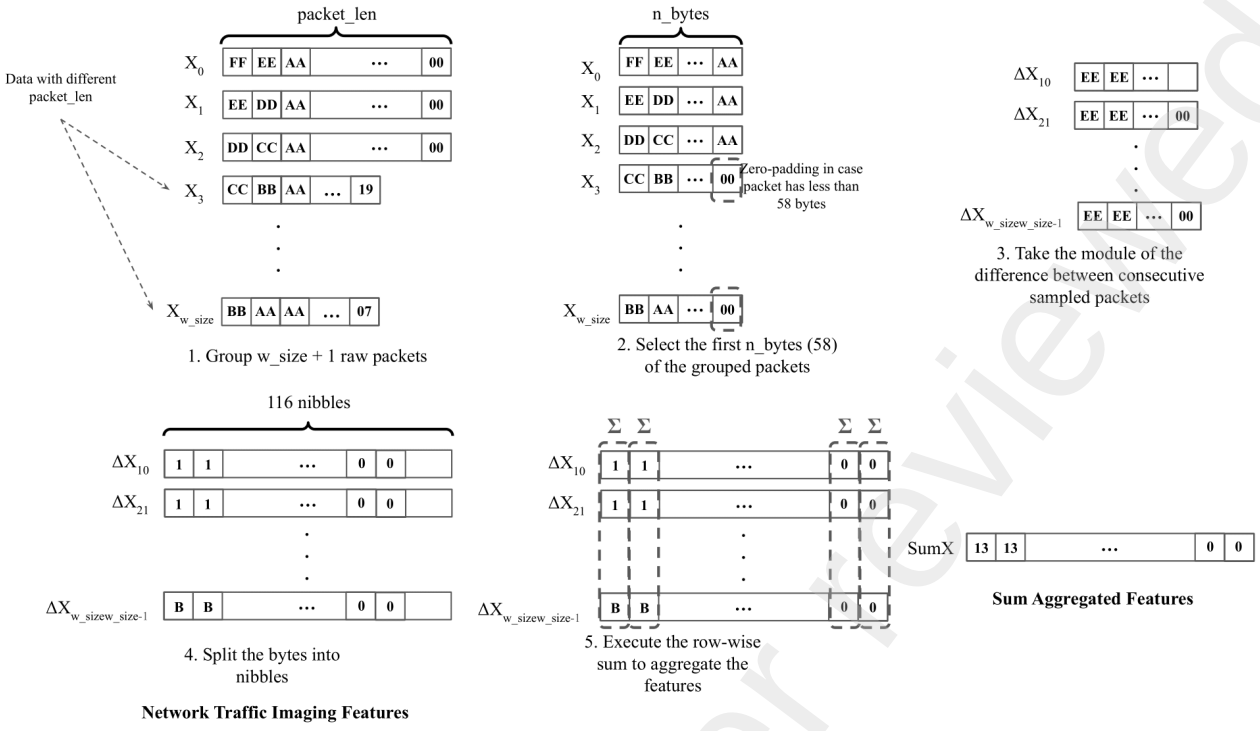


Figure 5: Steps of the proposed feature extractor. It is important to notice that this feature extractor has two outputs: the network traffic imaging features that will be fed to the Pruned CNN model and the sum aggregated features that the Random Forest model will use.

Table 3
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

4.3. Second stage: Pruned Convolutional Neural Network classifier

For the second stage model of our proposed IDS, we have chosen to use the Pruned CNN architecture from our previous work Luz et al. (2023). This decision was based on the fact that this architecture has shown promising results for the AEID dataset and has fewer parameters than the architecture presented in Jeong et al. (2021). The smaller number of parameters in our CNN architecture also improves the second-stage detection time metric.

We have only utilized the resulting Pruned CNN architecture obtained in Luz et al. (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 a first stage that focuses on lower detection time.

Moreover, we have also adapted the Pruned CNN model regarding its number of outputs. Since the TOW-IDS dataset Han et al. (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 second stage model for both datasets are presented in Table 4. 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. (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.

$$\texttt{out\_feat} = \begin{cases} 1, & \text{if dataset is AEID} \\ 6, & \text{if dataset is TOW-IDS} \end{cases}, \tag{9}$$

5. 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

Table 4
Pruned CNN architecture and hyperparameters obtained in Luz et al. (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. (9)
Output	Sigmoid	-	-

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 et al. (2023).

5.1. 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 et al. (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 et al. (2023); Han et al. (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 denial of service, 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 2, and the labeling criteria for the TOW IDS dataset can be found in Algorithm 3.

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 5.

Algorithm 2 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 3 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
```

Table 5
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%)

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 6 presents the final class distribution. Our distribution differs from the one presented in Han et al. (2023) since the distribution shown in Han et al. (2023) refers to the class distribution per sample without considering any grouping in the data. We believe this is a

Table 6
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%)

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.

5.2. 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. 6 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 first-stage Random Forest classifier and another for the second-stage Pruned CNN classifier.

5.3. 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 (10)-(15):

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}, \tag{10}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{11}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{12}$$

$$\text{F1-score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}, \tag{13}$$

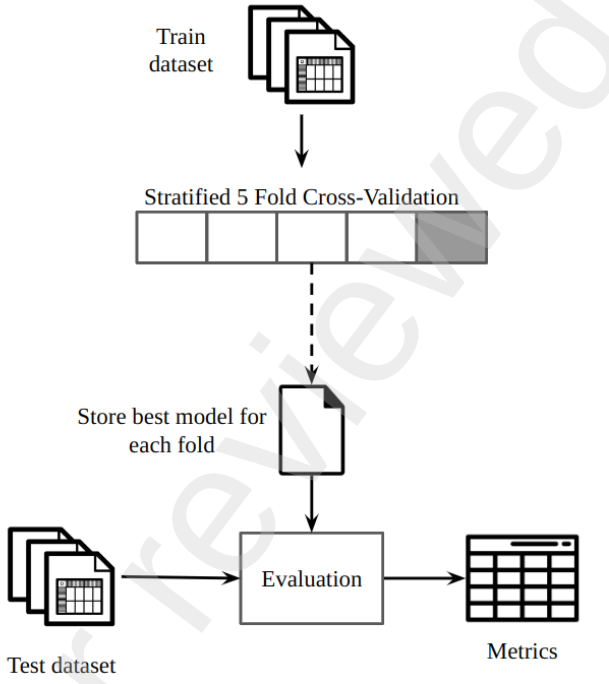


Figure 6: Methodology used for training, validating, and testing our proposed IDS.

$$\text{ROC AUC} = \int_0^1 TPR(FPR)\, dFPR, \tag{14}$$

and

$$\text{DT} = \frac{\text{Mean detection time}}{\text{Batch size}} \left[\frac{\mu s}{sample}\right], \tag{15}$$

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. (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:

$$\begin{aligned}\text{DT}_{\text{overall}} = \text{DT}_{\text{firststage}} \cdot \text{Acc}_{\text{firststage}} + \\ \text{DT}_{\text{secondstage}} \cdot (\text{Acc}_{\text{secondstage}} - \text{Acc}_{\text{firststage}}),\end{aligned} \tag{16}$$

where $DT_{firststage}$ and $DT_{secondstage}$ are the detection time for the first and second stage, respectively, and $Acc_{firststage}$ and $Acc_{secondstage}$ the Accuracy for the first and second stage, respectively.

6. 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. (2) and (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 et al. (2023) and two other works that propose IDSs but using the TOW-IDS dataset Han et al. (2023) and Shibly et al. (2023).

6.1. 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 first stage detected 99.7% of the packets correctly, but it also presented a high false positive rate compared to the second stage. Moreover, in the second 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 first stage was based on a simpler model and feature, it could still achieve significant detection results. The first-stage detection results show that for attacks such as the replay attacks in the AEID dataset, a traditional machine learning-based IDS can provide fast detection time without sacrificing its high detection results. The confusion matrix for the first and second stages of our proposed IDS for the AEID dataset are presented in Figs. 7a and 7b.

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 second-stage results, we see that the second-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 first and second stages of our proposed IDS for the TOW dataset can be seen in Figs. 7c and 7d, 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. 8a. As can be seen, the thresholds did not significantly impact the results of both the first and second 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. 8b, it is possible to see the performance decay in the first stage model. As we previously discussed, this is mainly because the model is unable to distinguish, as well as the second stage, the attacks present in the TOW-IDS dataset.

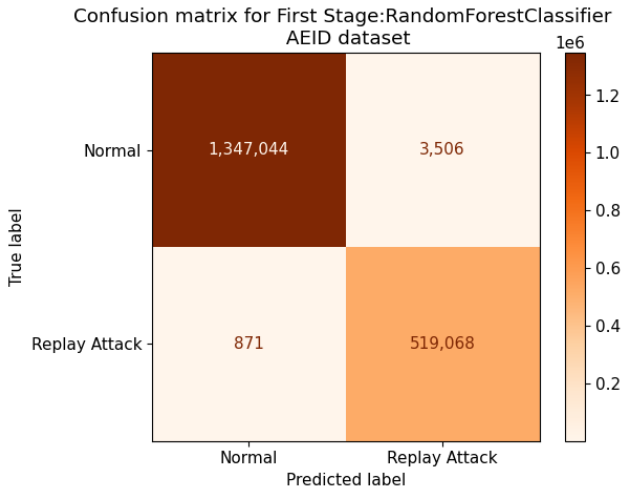
We compared our proposed IDS to the state-of-the-art works regarding their detection results: Accuracy, Precision, Recall, F1-Score, and ROC-AUC values. For the AEID dataset, we have obtained the first and second-best results with our second and first-stage models, respectively. The comparison results with the works that used the AEID dataset are presented in Table 7. 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 and 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).

Moreover, for the TOW-IDS dataset, as previously mentioned, there was a significant difference between our first and second-stage detection results, as can be seen in Table 8. The main explanation behind this difference is the network heterogeneity presented in this dataset. Despite the results of the first stage, we achieved the second-best detection results among the compared works, differing from the method proposed on Han et al. (2023) only in the third decimal digit. The second-stage detection metrics for the TOW-IDS dataset show that our multi-stage technique provides the expected results since the second stage's primary goal is to achieve higher detection results at a lower detection time.

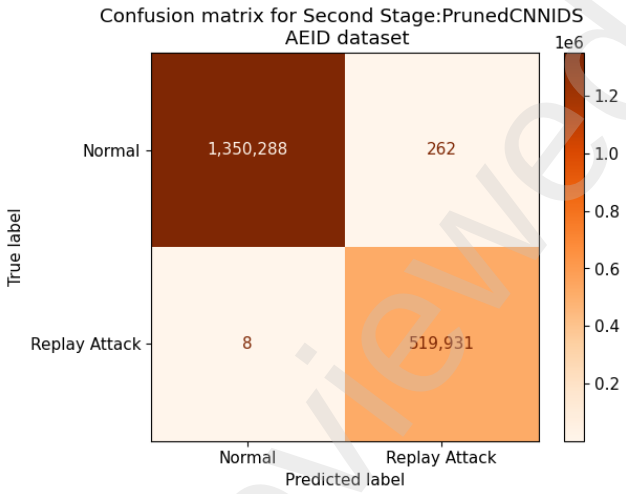
Finally, our accuracy modeling criteria defined in Eq. (2) were satisfied based on our experimental results. Surprisingly, for the AEID dataset, the difference in detection results between the first and second stages was only on the third decimal digit. This shows that our first stage can handle more straightforward attacks, such as the replay attacks present in the AEID dataset. Moreover, we have observed a difference in the first decimal difference between the first and second-stage models for the TOW-IDS dataset. Although this result still fulfills our modeling criteria, it shows room for improvement in the first stage model for more complex datasets and cyberattacks.

6.2. Detection time results

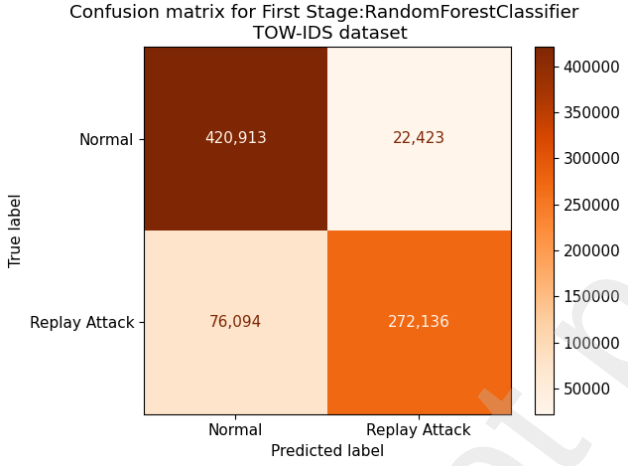
For the detection time results, we have used Eq. (15) with a batch size of 64 and considering the mean value of 500 samples.



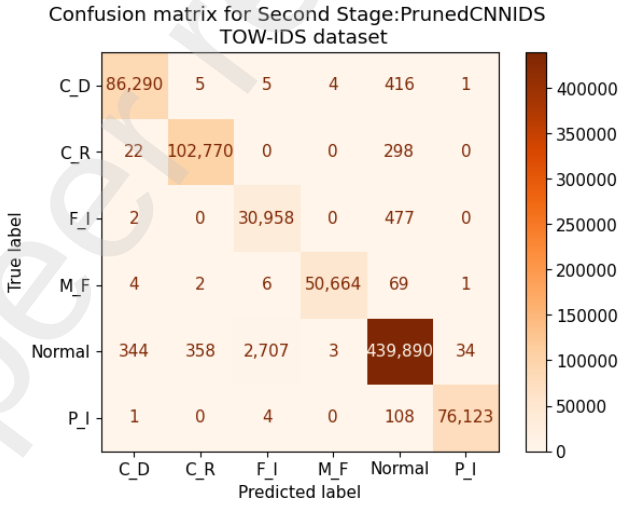
(a) First stage confusion matrix for AEID dataset.



(b) Second stage confusion matrix for AEID dataset.

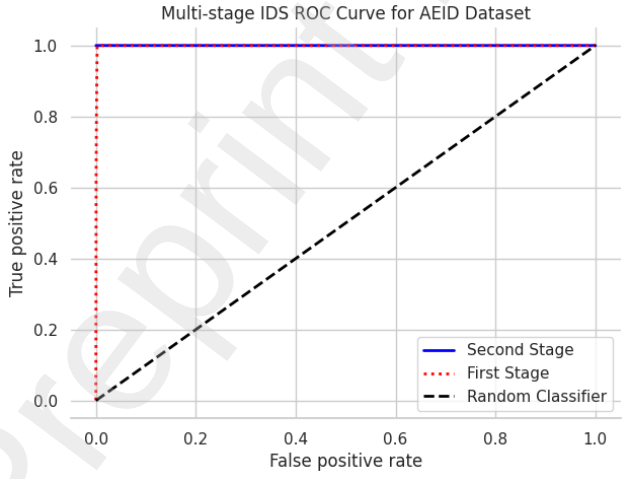


(c) First stage confusion for TOW-IDS dataset.

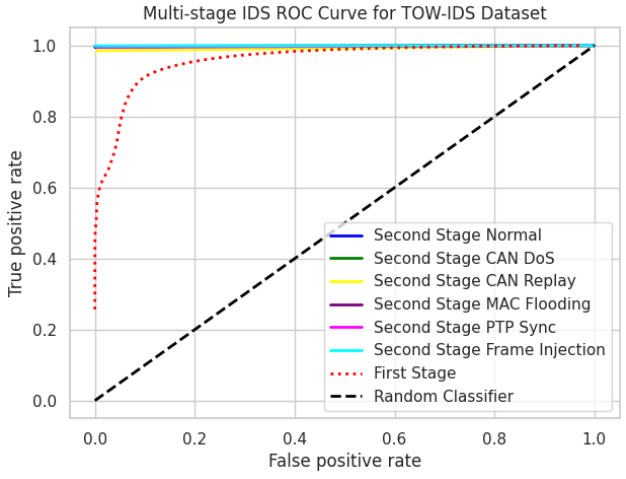


(d) Second stage confusion matrix for TOW-IDS dataset.

Figure 7: 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) ROC Curve of IDS stages for AEID dataset.



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

Figure 8: ROC Curve of our proposed IDS stages in both evaluated datasets.

Table 7
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 (First stage)	0.9976	0.9932	0.9982	0.9957	0.9993
Our work (Second stage)	**0.9997**	**0.9995**	**0.9997**	**0.9996**	**0.9998**
Luz et al. (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

Table 8
Test set results for TOW-IDS dataset. We have used the results presented in the original paper for both Han et al. (2023) and Shibly et al. (2023) works.

Work	Accuracy	Precision	Recall	F1-Score
Our work (First stage)	0.8740	0.9238	0.7778	0.8446
Our work (Second stage)	0.9962	0.9960	0.9962	0.9960
Han et al. (2023)	**0.9965**	-	-	**0.9974**
Shibly et al. (2023)	0.9700	0.9400	0.9500	0.9500

We have assessed the detection time results for both stages of our proposed IDS. Our detection time results are presented in Table 9. As expected, the first 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 second-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).

Although it is not possible to make a direct comparison between our results and the results presented in Han et al. (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 9, even with a worse computing platform, we have obtained a better result with our first 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 first-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 et al. (2023).

We carried out an overall detection time analysis of our proposed IDS by applying Eq. (16) with the results presented in Tables 7, 8 and 9. Therefore, we obtained an $DT_{overall}$ 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.

At last, our IDS timing criteria proposed in (1) were successfully satisfied and validated by our experimental results. We have obtained a $DT_{firststage}$ 42 times smaller than the $DT_{secondstage}$ for the AEID dataset models and 21 times smaller for the TOW IDS dataset.

7. Conclusion and Future Works

In this work, we proposed a novel multi-stage deep learning approach that consists of two stages. The first stage aims to ensure a fast detection time, while the second stage focuses on achieving the most accurate results. To achieve this, we have used a Random Forest classifier in the first stage and a Pruned CNN, as obtained in our previous work Luz et al. (2023), in the second 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 et al. (2023). We have also compared our experimental results with state-of-the-art works, and our second-stage and first-stage 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 et al. (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

Table 9
Detection time metrics for the compared works. *: 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 (us/sample)	
		AEID dataset	TOW-IDS dataset
Intel(R) Xeon(R) CPU @ 2.20GHz	Our work (First stage)	**107**	**221**
	Our work (Second stage)	4510	4757
	Luz et al. (2023)*	4510	-
	Carmo et al. (2022)*	383	-
	Jeong et al. (2021)*	7200	-
4790K CPU, 32GB RAM, and 2080 RTX GPU	Han et al. (2023)**	-	250

detection threshold of 1,735 μs/sample proposed in Jeong et al. (2021).

In our future work, we plan to introduce a feedback branch that connects the first-stage and second-stage models to improve the results of the first stage. Our experimental results have shown that conventional machine learning ensemble models, such as Random Forest and XGBoost Carmo et al. (2022), can balance a fast detection time while maintaining significant detection results. Hence, we aim to leverage the second stage in the learning process to enhance the detection results of the first stage.

One potential direction is to develop an unsupervised intrusion detection system to address the drawback of our proposed IDS, which is its inability to detect zero-day attacks. We also aim to offer new cyberattacks to the automotive network environment using an automotive Ethernet testbed built with industry hardware to experiment and create new datasets.

Acknowledgement

This work was supported in part by the Microsoft Research Ph.D. Fellowship and by CNPq (Grant 312368/2021-6).

References

Alkhatib, N., Ghauch, H., Danger, J.L., 2021. 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), pp. 0954–0962. doi:10.1109/IEMCON53756.2021.9623129.

Alkhatib, N., Mushtaq, M., Ghauch, H., Danger, J.L., 2022. Unsupervised network intrusion detection system for avtp in automotive ethernet networks, in: 2022 IEEE Intelligent Vehicles Symposium (IV), IEEE Press. p. 1731–1738. URL: https://doi.org/10.1109/IV51971.2022.9827285, doi:10.1109/IV51971.2022.9827285.

Freitas de Araujo-Filho, P., Kaddoum, G., Campelo, D.R., Santos, A.G., Macêdo, D., Zanchettin, C., 2020. Intrusion detection for cyber–physical systems using generative adversarial networks in fog environment. IEEE Internet of Things Journal 8, 6247–6256.

Bianco, S., Cadene, R., Celona, L., Napoletano, P., 2018. Benchmark analysis of representative deep neural network architectures. IEEE Access 6, 64270–64277. doi:10.1109/ACCESS.2018.2877890.

Breiman, L., 2001. Random forests. Machine learning 45, 5–32.

Carmo, P., Freitas de Araujo-Filho, P., Campelo, D., Freitas, E., Filho, A.O., Sadok, D., 2022. 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, SBC, Porto Alegre, RS, Brasil. pp. 196–209. URL: https://sol.sbc.org.br/index.php/sbrc/article/view/21171, doi:10.5753/sbrc.2022.222153.

Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T., 2011. Comprehensive experimental analyses of automotive attack surfaces, in: 20th USENIX Security Symposium (USENIX Security 11), USENIX Association, San Francisco, CA. URL: https://www.usenix.org/conference/usenix-security-11/comprehensive-experimental-analyses-automotive-attack-surfaces.

Daş, R., Karabade, A., Tuna, G., 2015. Common network attack types and defense mechanisms, in: 2015 23nd Signal Processing and Communications Applications Conference (SIU), IEEE. pp. 2658–2661.

Frankle, J., Carbin, M., 2018. The lottery ticket hypothesis: Training pruned neural networks. CoRR abs/1803.03635. URL: http://arxiv.org/abs/1803.03635, arXiv:1803.03635.

Freitas De Araujo-Filho, P., Pinheiro, A.J., Kaddoum, G., Campelo, D.R., Soares, F.L., 2021. An Efficient Intrusion Prevention System for CAN: Hindering Cyber-attacks with a Low-cost Platform. IEEE Access , 1–1doi:10.1109/access.2021.3136147.

Ghosal, A., Conti, M., 2020. Security issues and challenges in V2X : A Survey. Computer Networks 169, 107093. URL: https://doi.org/10.1016/j.comnet.2019.107093, doi:10.1016/j.comnet.2019.107093.

Han, M.L., Kwak, B.I., Kim, H.K., 2023. TOW-IDS: Intrusion detection system based on three overlapped wavelets for automotive ethernet. IEEE Transactions on Information Forensics and Security 18, 411–422. doi:10.1109/TIFS.2022.3221893.

IEEE, 2016. 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) , 1–233doi:10.1109/IEEESTD.2016.7782716.

IEEE, 2020. 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) , 1–421doi:10.1109/IEEESTD.2020.9121845.

Jeong, S., Jeon, B., Chung, B., Kim, H.K., 2021. Convolutional neural network-based intrusion detection system for AVTP streams in automotive ethernet-based networks. Vehicular Communications 29, 100338. URL: https://www.sciencedirect.com/science/article/pii/S2214209621000073, doi:https://doi.org/10.1016/j.vehcom.2021.100338.

Jeong, S., Kim, H.K., Han, M.L., Kwak, B.I., 2023. AERO: Automotive ethernet real-time observer for anomaly detection in in-vehicle networks. IEEE Transactions on Industrial Informatics , 1–12doi:10.1109/TII.2023.3324949.

Jo, H.J., Choi, W., 2021. A Survey of Attacks on Controller Area Networks and Corresponding Countermeasures. IEEE Transactions on Intelligent Transportation Systems , 1–19doi:10.1109/TITS.2021.3078740.

Lai, C., Lu, R., Zheng, D., Shen, X., 2020a. Security and privacy challenges in 5g-enabled vehicular networks. IEEE Network 34, 37–45. doi:10.1109/MNET.001.1900220.

Lai, C., Lu, R., Zheng, D., Shen, X., 2020b. Security and privacy challenges in 5g-enabled vehicular networks. IEEE Network 34, 37–45. doi:10.1109/MNET.001.1900220.

Lampe, B., Meng, W., 2023. can-train-and-test: A curated can dataset for automotive intrusion detection. arXiv:2308.04972.

Liu, J., Zhang, S., Sun, W., Shi, Y., 2017. In-vehicle network attacks and countermeasures: Challenges and future directions. IEEE Network 31, 50–58. doi:10.1109/MNET.2017.1600257.

Luz, L., Freitas de Araujo-Filho, P., Campelo, D., 2023. 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, Porto Alegre, RS, Brasil. pp. 197–210. URL: https://sol.sbc.org.br/index.php/sbrc/article/view/24539, doi:10.5753/sbrc.2023.527.

Matheus, K., Königseder, T., 2021. Automotive Ethernet. Cambridge University Press.

Miller, C., Valasek, C., 2015. Remote Exploitation of an Unaltered Passenger Vehicle. Defcon 23 2015, 1–91. URL: http://illmatics.com/RemoteCarHacking.pdf.

Moussa, B., Kassouf, M., Hadjidj, R., Debbabi, M., Assi, C., 2019. An extension to the precision time protocol (PTP) to enable the detection of cyber attacks. IEEE Transactions on Industrial Informatics 16, 18–27.

Nisioti, A., Mylonas, A., Yoo, P.D., Katos, V., 2018. From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. IEEE Communications Surveys & Tutorials 20, 3369–3388.

Seo, E., Song, H.M., Kim, H.K., 2018. GIDS: GAN based intrusion detection system for in-vehicle network, in: 2018 16th Annual Conference on Privacy, Security and Trust (PST), pp. 1–6. doi:10.1109/PST.2018.8514157.

Shibly, K.H., Hossain, M.D., Inoue, H., Taenaka, Y., Kadobayashi, Y., 2023. A feature-aware semi-supervised learning approach for automotive ethernet, in: 2023 IEEE International Conference on Cyber Security and Resilience (CSR), pp. 426–431. doi:10.1109/CSR57506.2023.10224976.

Song, H.M., Woo, J., Kim, H.K., 2020. In-vehicle network intrusion detection using deep convolutional neural network. Vehicular Communications 21, 100198.

Sun, X., Yu, F.R., Zhang, P., 2022. A survey on cyber-security of connected and autonomous vehicles (CAVs). IEEE Transactions on Intelligent Transportation Systems 23, 6240–6259. doi:10.1109/TITS.2021.3085297.

Tuohy, S., Glavin, M., Hughes, C., Jones, E., Trivedi, M., Kilmartin, L., 2015. Intra-Vehicle Networks: A Review. IEEE Transactions on Intelligent Transportation Systems 16, 534–545. doi:10.1109/TITS.2014.2320605.

UN Regulation 155, 2021. UN Regulation No. 155 - cyber security and cyber security management system. 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., 2022. IEEE 802.1AS clock synchronization performance evaluation of an integrated wired–wireless tsn architecture. IEEE Transactions on Industrial Informatics 18, 2986–2999. doi:10.1109/TII.2021.3106568.

Verma, M.E., Iannacone, M.D., Bridges, R.A., Hollifield, S.C., Moriano, P., Kay, B., Combs, F.L., 2022. Addressing the lack of comparability & testing in can intrusion detection research: A comprehensive guide to can ids data & introduction of the road dataset. arXiv:2012.14600.

Wu, W., Li, R., Xie, G., An, J., Bai, Y., Zhou, J., Li, K., 2020. A survey of intrusion detection for in-vehicle networks. IEEE Transactions on Intelligent Transportation Systems 21, 919–933. doi:10.1109/TITS.2019.2908074.

Yang, L., Moubayed, A., Hamieh, I., Shami, A., 2019. Tree-based intelligent intrusion detection system in internet of vehicles, in: 2019 IEEE Global Communications Conference (GLOBECOM), pp. 1–6. doi:10.1109/GLOBECOM38437.2019.9013892.

Luigi F. Marques da Luz received the bachelor's degree in electronic engineering from the Universidade Federal Rural de Pernambuco (UFRPE), Cabo de Santo Agostinho, Brazil, in 2021. He is currently pursuing an M.Sc. degree in computer science with UFPE. He is a Computer Engineer with the Centro de Estudos e Sistemas Avançados do Recife (CESAR), working with embedded systems, the Internet of Things, and artificial intelligence applications. His research interests include in-vehicle network security, intrusion detection systems, and artificial intelligence.

Paulo Freitas de Araujo-Filho received the bachelor's degree (with Hons.) in electrical and electronic engineering and the M.S. and Ph.D. degrees in computer science from the Universidade Federal de Pernambuco (UFPE), Recife, Brazil, in 2016, 2018, and 2023, respectively. He also received the Ph.D. degree in electrical engineering from the École de Technologie Supérieure, Université du Québec, Montreal, QC, Canada, in 2023. He works as an Assistant Professor at UFPE. His current research interests include security, intrusion detection systems, artificial intelligence, Internet of Things, and adversarial attacks.

Divanilson R. Campelo is an Associate Professor of Computer Science and Computer Engineering at the Centro de Informática (CIn), Universidade Federal de Pernambuco (UFPE), Recife, Brazil. He received the degree in electrical engineering from UFPE, Recife, Brazil, in 1998, and the M.Sc. and Doctoral degrees in electrical engineering from the Universidade Estadual de Campinas (UNICAMP), Campinas, Brazil, in 2001 and 2006, respectively. He was a Visiting Assistant Professor with the Electrical Engineering Department, Stanford University, Stanford, CA, USA, from September 2008 to December 2009. His recent research interests include automotive and vehicular networking, connected objects, broadband access networks and cybersecurity.