Multi-Criteria Optimized Deep Learning-based Intrusion Detection System for Detecting Cyberattacks in Automotive Ethernet Networks

Luigi F. Marques da Luz[1,2], **Paulo Freitas de Araujo-Filho**[1,3], **Divanilson R. Campelo**[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (CIn - UFPE)
Av. Jorn. Aníbal Fernandes – s/n – Recife – PE – Brazil

[2]Centro de Estudos e Sistemas Avançados do Recife (CESAR)
Rua Bione - 220 - Recife - PE - Brazil

[3]École de Technologie Supérieure, Université du Québec,
Montreal, QC, H3C 1K3, Canada

{lfml,pfaf,dcampelo}@cin.ufpe.br

Abstract. *Connected and autonomous vehicles (CAVs) are part of the Internet of Things, exposing them to cyberattacks. CAVs comprise several systems, such as advanced driver assistance systems, that require high bandwidth for critical data transmission, where automotive Ethernet plays an essential role as an enabling technology. In this paper, we propose a deep learning-based intrusion detection system for detecting replay attacks 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 900x in the storage size and a speedup of 1.4x in the detection time with a negligible drop in the F1-score compared to existing work.*

1. Introduction

Connected and autonomous vehicles (CAVs) have been playing a significant role in improving safety and reducing the costs of the transportation segment. CAVs have advanced driver assistance systems (ADAS) that will enhance the drivers' user experience, safety, and security by introducing features such as parking assistance, pedestrian detection, collision avoidance, and adaptive cruise control [Kukkala et al. 2018]. LIDARs and camera sensors are essential to improve the vehicle's perception system. Such sensors require higher bandwidths than the automotive networks such as controller area networks (CAN) can provide [Matheus and Königseder 2021].

Trends in the automotive industry require a flexible, scalable, and future-proof in-vehicle network technology to address the rapidly changing customer expectations [Matheus and Königseder 2021]. As automotive Ethernet provides higher bandwidths, it is an enabling technology for autonomous driving applications. Precisely, automotive Ethernet comprehends several standards to meet different bandwidth requirements. For instance, IEEE 100BASE-T1 provides a 100 Mbps operation over a single twisted pair balanced cabling, while CAN and FlexRay provide only 500 Kbps and 10 Mbps of data rate, respectively [Matheus and Königseder 2021, Bandur et al. 2021].

Moreover, automotive Ethernet also comprehends a set of standards that compose audio-video bridging (AVB) and time-sensitive networking (TSN) [Tuohy et al. 2015]. AVB comprises standards that provide improved synchronization, low latency, and reliability for Ethernet networks. On the other hand, TSN defines mechanisms for the time-sensitive transmission of control data over deterministic Ethernet networks. Among those standards, 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 for CAVs [IEEE 2016].

CAVs enable several applications, such as transportation efficiency enhancement, safety improvement, and alleviation of environmental damage due to their connectivity to the outside world [Sun et al. 2022]. However, this comes with their exposure to cyberattacks that may harm drivers, passengers, and pedestrians [Koscher et al. 2010, Liu et al. 2017, Jo and Choi 2021, Miller and Valasek 2015, Ghosal and Conti 2020]. Therefore, it is essential to defend connected vehicles against such threats.

Encryption and authentication are effective ways of guaranteeing security, but they are usually unsuitable for in-vehicle networks (IVN). IVN environments are cost-constrained, require real-time reliability, and have limited computing capacity and storage resources. Intrusion detection systems (IDSs) have shown promising results in detecting attacks in these scenarios [Wu et al. 2020]. IDSs work as a second line of defense and the last resort when other security mechanisms fail. As a matter of fact, [UN Regulation 2021] establishes that commercial vehicles must be able to detect cyberattacks, making intrusion detection capability a requirement for vehicles manufactured after July 2022.

Meanwhile, deep learning (DL) techniques have been adopted in IDSs because of their ability to learn hidden patterns in complex data, thus presenting promising results [Lansky et al. 2021, Jeong et al. 2021]. On the other hand, DL models usually demand high computational power and storage size due to the many inner connections and layers of these models, increasing the cost of applications that use this kind of technology [Bianco et al. 2018]. Hence, there is still room for improvement to ease the adoption of DL-based IDSs in resource-constrained environments such as IVNs, once the cost is one of the major concerns of the automotive industry [Wu et al. 2020].

Therefore, in this paper, we propose a DL-based IDS that uses a loss function that simultaneously improves storage size, detection time, and detection results during the model training step, generating an optimized model regarding its storage size and detection time. We aim to shorten the gap between deploying DL-based IDSs in resource-constrained environments. This optimization occurs by inducing a lower bit representation and sparsity in the neural network's weights during the training process without requiring a post-training optimization process such as model quantization or pruning.

Our application scenario will distinguish benign or malicious AVTP packets in an automotive Ethernet network, specifically replay attacks. To perform this attack, the intruder must have pre-captured packets that will be reinjected into the network to confuse the network nodes that rely on this information. In this scenario, malicious packets may put in danger the life of the driver and people around them when the injected information

is one of the sources of the decision-making process of the CAV. In a nutshell, the main contributions of this work are:

- We 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. By doing so, we shorten the gap between deploying DL-based IDSs in resource-constrained environments such as an automotive network.
- An experimental comparison between the existing works. This comparison shows a reduction of 900x and 1.43x, respectively, in the storage size and detection time compared to the method presented in [Jeong et al. 2021] while maintaining similar results regarding the F1-score.

This paper is organized as follows. In Section 2, we present the recent advances in automotive intrusion detection systems. Section 3 introduces the architecture of our proposed IDS and describes its optimization technique. Section 4 explains the experimental setup and methodology used to evaluate our IDS. In Section 5, we show the results of IDS and compare them to those of state-of-the-art automotive Ethernet IDSs. Finally, Section 6 concludes our paper and discusses future works.

2. Related Works

Given its challenges, automotive network intrusion detection has been an active research field in recent years. Hence, the work in [Freitas De Araujo-Filho et al. 2021] presented an intrusion prevention system to detect and prevent unknown cyberattacks before receiving a CAN frame. Similarly, the work in [Seo et al. 2018] relies on a generative adversarial network that can detect zero-day attacks in a CAN network only using normal traffic data.

As for the works considering intrusion detection on automotive Ethernet, the authors of [Jeong et al. 2021] proposed an IDS based on a 2D-convolutional neural network (2D-CNN) for detecting replay attacks in AVTP packets. Their model achieved F1-score and recall values greater than 0.9704 and 0.9949, respectively. However, as their model relies on supervised learning, it requires labeled data, which is complicated and sometimes impossible to obtain and cannot detect unknown attacks. Moreover, it requires GPUs to achieve short detection times, incurring a high deployment cost.

On the other hand, the authors of [Carmo et al. 2022] relied on the XGBoost algorithm to detect replay attacks in AVTP packets. They have achieved Receiver Operator Characteristic Area Under Curve (ROC AUC) values of 0.9805 and a detection time of 620 µs/sample using low-cost CPU-based hardware, such as a Raspberry Pi. However, they do not evaluate the model's storage size, so it is unclear whether their IDS can run on microcontroller devices, such as those used in ECUs.

In [Alkhatib et al. 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. They achieved an F1-score of 0.98, an detection time of 0.45 seconds, and a 101 MB storage size for the CAE model, while for LSTM-AE, they obtained an F1-score around 0.80, a detection time of 2.70 seconds, and a storage size of 52 KB.

In [Alkhatib et al. 2021], the authors evaluated the performance of DL-based IDSs for detecting cyberattacks in automotive Ethernet, obtaining F1-score and ROC AUC values greater than 0.8. 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.

In contrast to the existing works, our work proposes a technique to simultaneously optimize its detection-related metrics (such as accuracy, precision, recall, F1-score, and ROC AUC), timing requirements, and storage size during the training step without being directly attached to a model architecture. This points toward shortening the gap between deploying DL-based IDSs in resource-constrained environments such as ECUs.

3. Proposed System

In this section, we propose a DL-based IDS whose primary goal is to detect replay cyber-attacks 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 3.3) 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 1). 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.

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 1 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.

3.1. 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

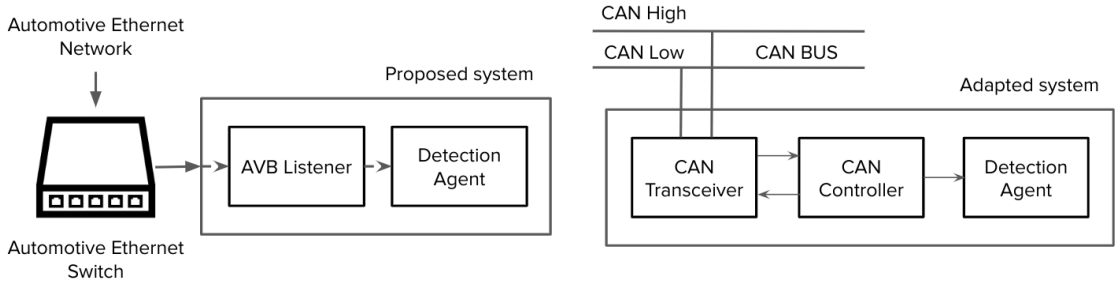

Figure 1. On the left is our proposed IDS architecture to be deployed in an automotive Ethernet environment. On the right is how our architecture could be adapted to be deployed in a CAN bus environment.

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 3.2) 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
```

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 2. 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.

3.2. Feature generator

For the feature generator, we have chosen to maintain the one initially proposed in [Jeong et al. 2021], which will be 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 will 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 3.

Figure 2. 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. Adapted from [Burke 2019].

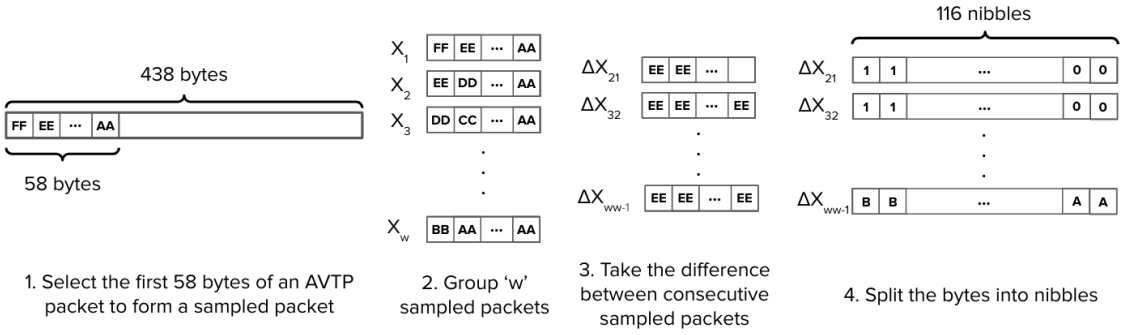

Figure 3. The steps of the feature generator.

3.3. 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:

$$\mathcal{L}(\Theta) = \mathcal{L}(\Theta)_{detection} + \mathcal{L}(\Theta)_{storage\ size} + \mathcal{L}(\Theta)_{detection\ time}, \quad (1)$$

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 4.

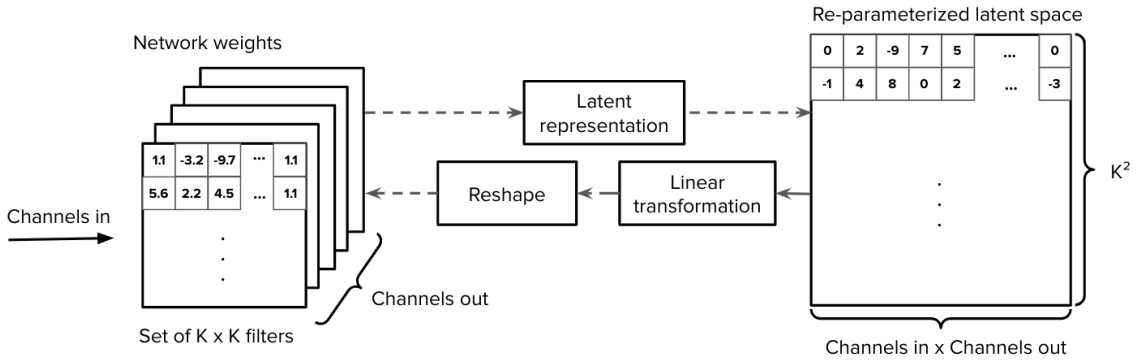

Figure 4. The transformation process of network weights shape during the LilNetX framework optimization.

3.4. 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 1. The main differences in our architecture are regarding 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 1. 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	-	Dropout	in_feat=20416, out_feat=64	Dense
Dense	ReLU	Dropout	in_feat=64, out_feat=1	Dense
Output	Sigmoid	-	-	-

4. Methodology and Experimental Evaluation

In this section, we provide 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.

4.1. 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 5. 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 3.2). At last, the features and labels are combined to generate the prepared dataset. This preparation process is executed once for indoor and outdoor data.

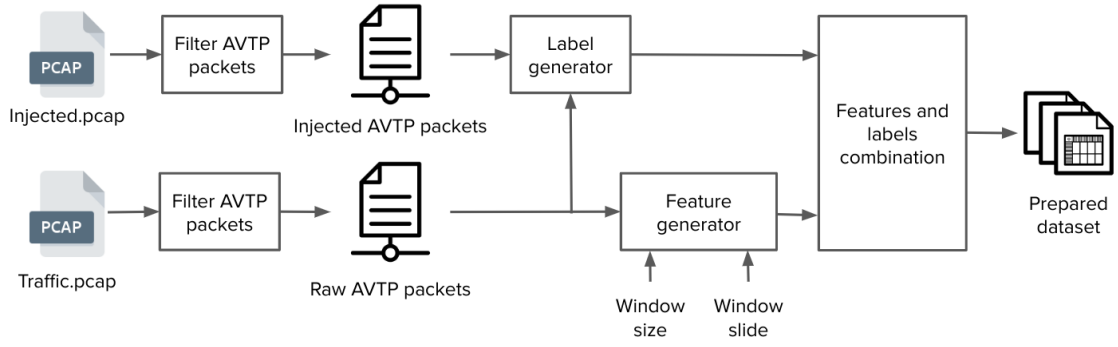

Figure 5. AVTP Intrusion dataset preparation process.

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.

4.2. 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 6.

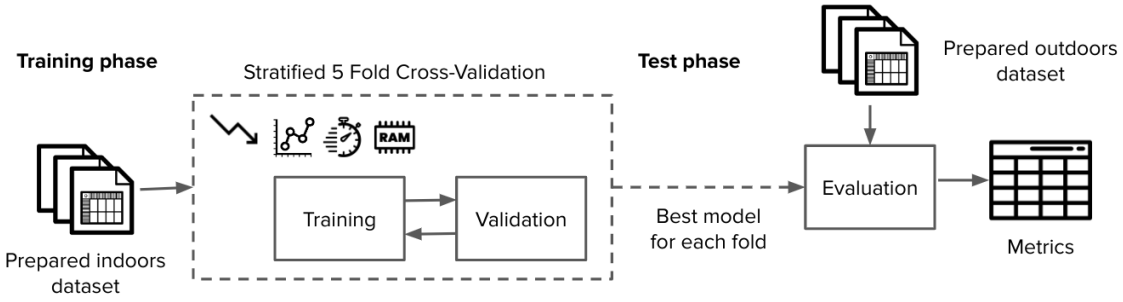

Figure 6. Methodology of training and evaluation proposed multi-criteria optimized automotive Ethernet IDS.

5. 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.

5.1. 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 2.

As seen in Table 2, 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

Table 2. 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

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 3 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-Model, which uses an ensemble tree-based machine learning model and benefits from the low detection time of tree-based models.

Table 3. 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

Table 3 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.

5.2. 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

4. 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 4. Storage size metrics comparison.

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

Table 5. Detection time metrics comparison.

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

5.3. 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 5. 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 1) 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 timing consuming computation.

5.4. Trade-off analysis

Finally, we analyzed the trade-off between the three evaluated DL-based IDSs for automotive Ethernet. Table 6 summarizes their F1-score, detection time, and storage size. The detection time and storage size in Table 6 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 6. 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

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. Conclusions and Future Works

This work 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.

We have also analyzed the trade-off between detection results, detection time, and storage size. The balance of the abovementioned metrics is essential to design 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.

Malicious attackers create new cyberattacks constantly, and IDSs must be able to detect these kinds of threats. For future work, we aim to develop IDSs that detect zero-day attacks. In this direction, unsupervised deep learning models such as autoencoders [Alkhatib et al. 2022] have been shown as a suitable technique for detecting unknown attacks and still have room for optimization to meet the detection time and storage size requirements for automotive networks. We also intend to conduct experiments with hard-

ware equipment used by the automotive industry.

References

[Alkhatib et al. 2021] Alkhatib, N., Ghauch, H., and Danger, J.-L. (2021). SOME/IP Intrusion Detection using Deep Learning-based Sequential Models in Automotive Ethernet Networks.

[Alkhatib et al. 2022] Alkhatib, N., Mushtaq, M., Ghauch, H., and Danger, J.-L. (2022). Unsupervised Network Intrusion Detection System for AVTP in Automotive Ethernet Networks.

[Bandur et al. 2021] Bandur, V., Selim, G., Pantelic, V., and Lawford, M. (2021). Making the case for centralized automotive e/e architectures. *IEEE Transactions on Vehicular Technology*, 70(2):1230–1245.

[Bianco et al. 2018] Bianco, S., Cadene, R., Celona, L., and Napoletano, P. (2018). Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277.

[Burke 2019] Burke, K. (2019). How does a self-driving car see? `https://blogs.nvidia.com/blog/2019/04/15/how-does-a-self-driving-car-see/`. Accessed: 2022-12-30.

[Carmo et al. 2022] Carmo, P., Araujo-Filho, P., Campelo, D., Freitas, E., Filho, A. O., and 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*, pages 196–209, Porto Alegre, RS, Brasil. SBC.

[Freitas De Araujo-Filho et al. 2021] Freitas De Araujo-Filho, P., Pinheiro, A. J., Kaddoum, G., Campelo, D. R., and Soares, F. L. (2021). An Efficient Intrusion Prevention System for CAN: Hindering Cyber-attacks with a Low-cost Platform. *IEEE Access*, pages 1–1.

[Ghosal and Conti 2020] Ghosal, A. and Conti, M. (2020). Security issues and challenges in V2X : A Survey. *Computer Networks*, 169:107093.

[Girish et al. 2022] Girish, S., Gupta, K., Singh, S., and Shrivastava, A. (2022). Lilnetx: Lightweight networks with extreme model compression and structured sparsification.

[IEEE 2016] 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)*, pages 1–233.

[Jeong et al. 2021] Jeong, S., Jeon, B., Chung, B., and Kim, H. K. (2021). Convolutional neural network-based intrusion detection system for AVTP streams in automotive Ethernet-based networks. *Vehicular Communications*, 29:100338.

[Jo and Choi 2021] Jo, H. J. and Choi, W. (2021). A Survey of Attacks on Controller Area Networks and Corresponding Countermeasures. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–19.

[Koscher et al. 2010] Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., et al. (2010). Experi-

mental security analysis of a modern automobile. In *2010 IEEE symposium on security and privacy*, pages 447–462. IEEE.

[Kukkala et al. 2018] Kukkala, V. K., Tunnell, J., Pasricha, S., and Bradley, T. (2018). Advanced driver-assistance systems: A path toward autonomous vehicles. *IEEE Consumer Electronics Magazine*, 7(5):18–25.

[Lansky et al. 2021] Lansky, J., Ali, S., Mohammadi, M., Majeed, M. K., Karim, S. H. T., Rashidi, S., Hosseinzadeh, M., and Rahmani, A. M. (2021). Deep learning-based intrusion detection systems: A systematic review. *IEEE Access*, 9:101574–101599.

[Liu et al. 2017] Liu, J., Zhang, S., Sun, W., and Shi, Y. (2017). In-vehicle network attacks and countermeasures: Challenges and future directions. *IEEE Network*, 31(5):50–58.

[Matheus and Königseder 2021] Matheus, K. and Königseder, T. (2021). *Automotive ethernet*. Cambridge University Press.

[Miller and Valasek 2015] Miller, C. and Valasek, C. (2015). Remote Exploitation of an Unaltered Passenger Vehicle. *Defcon 23*, 2015:1–91.

[Oktay et al. 2019] Oktay, D., Ballé, J., Singh, S., and Shrivastava, A. (2019). Scalable model compression by entropy penalized reparameterization. *arXiv preprint arXiv:1906.06624*.

[Seo et al. 2018] Seo, E., Song, H. M., and 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)*, pages 1–6.

[Sun et al. 2022] Sun, X., Yu, F. R., and Zhang, P. (2022). A survey on cyber-security of connected and autonomous vehicles (cavs). *IEEE Transactions on Intelligent Transportation Systems*, 23(7):6240–6259.

[Tuohy et al. 2015] Tuohy, S., Glavin, M., Hughes, C., Jones, E., Trivedi, M., and Kilmartin, L. (2015). Intra-Vehicle Networks: A Review. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):534–545.

[UN Regulation 2021] UN Regulation (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`.

[Wu et al. 2020] Wu, W., Li, R., Xie, G., An, J., Bai, Y., Zhou, J., and Li, K. (2020). A survey of intrusion detection for in-vehicle networks. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):919–933.