Machine Learning-Based Intrusion Detection System for Automotive Ethernet: Detecting Cyber-Attacks with a Low-Cost Platform

Pedro R. X. Carmo[1,2], Paulo Freitas de Araujo-Filho[1,3], Divanilson R. Campelo [1], Eduardo Freitas [1,2], Assis T. de Oliveira Filho [1,2], Djamel F. H. Sadok [1,2]

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

[2]Grupo de Pesquisas em Redes e Telecomunicações (GPRT)
Universidade Federal de Pernambuco (UFPE)
Recife, Pernambuco, Brazil

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

{pedro.carmo,eduardo.freitas,assis.tiago,jamel}@gprt.ufpe.br

{pfaf,dcampelo}@cin.ufpe.br

Abstract. *Automotive Ethernet is being adopted in vehicles to provide the larger throughput that is required by autonomous vehicles. However, these vehicles may be subject to several cyber-attacks that compromise their operation and passengers' safety. This work proposes an Intrusion Detection System (IDS) that detects stream injection attacks on automotive Ethernet networks. The proposed IDS is based on feature generation and the XGBoost machine learning algorithm. Experimental results show that our proposed IDS achieves 0.9805 of AUCROC and a detection time of 620μs that allows real-time intrusion detection while using an inexpensive hardware platform, such as a Raspberry Pi. This is extremely important as cost is one of the automotive industry's main concerns.*

1. Introduction

Connected and Autonomous Vehicles (CAVs) are the primary trend for the future of the automotive industry. Such vehicles offer several features that demand increased image and sensor capabilities, such as high-resolution cameras and video-on-demand, among others. These features demand higher bandwidth to provide safety, comfort, and infotainment applications to drivers and passengers [Bello 2011].

In this context, the emergence of automotive Ethernet, especially the 100BASE-T1 standard, as a new in-vehicle network solution opened a myriad of opportunities for the introduction of new functionalities in cars, mainly due to Ethernet's ability to carry data with high bandwidth [Porter 2018].

Moreover, the compatibility of Ethernet with TCP/IP has also enabled other applications in cars, including diagnostics over IP and location-based services. The Audio Video Bridging (AVB) set of standards, which were developed with the aim of enabling

reliable transport of Audio/Video applications over Ethernet, provide several specifications to facilitate Quality of Service (QoS) guarantees for streaming data in an Ethernet-based in-vehicle network. Such specifications are defined in a set of standards, which jointly allow Ethernet to support end-point synchronization, timing, bounded latency, and bandwidth allocation, among other attributes [Matheus and Königseder 2021].

Since CAVs can be viewed as part of the Internet of Things (IoT), they are also vulnerable to cyberattacks that may put drivers, passengers, and pedestrians at risk [Liu et al. 2017], [El-Rewini et al. 2020], [Wu et al. 2020], [Koscher et al. 2010]. Intrusion Detection Systems (IDSs) for in-vehicle networks, which are a resource used when other countermeasures against cyber-physical attacks fail, must fulfill requirements such as real-time execution and high precision [Wu et al. 2020]. In general, since vehicle's electronic control units (ECUs) have limited resources and cost is one of the major concerns of the automotive industry, using a high-cost GPU to run IDSs in vehicles is not recommended. So it is desirable that IDSs in cars be capable of being deployed on low-cost hardware. However, a challenge arises in proposing an algorithm for an IDS that is complex enough to capture the details involved in the identification of benign and malicious packets, and at the same time is not computationally expensive to detect intrusions in real-time using inexpensive hardware.

In this direction, algorithms that use Gradient Boosting (GB) are robust classifiers that work well on structured data [Chen and Guestrin 2016]. In general, GB algorithms are good choices as they are easy to implement and work well even with minimal tweaks. The training time is fast and, unlike Deep Learning models, it is possible to obtain good performance with little training time in GB algorithms. GB uses ensemble learning, which combines classification from multiple base learners, learning complex relationships between resources. Furthermore, it is implemented using binary decision trees, which are great for learning non-parametric and non-linear relationships between features and targets.

In this work, an IDS based on the XGBoost machine learning classifier is proposed to identify replay attacks in automotive Ethernet networks. This type of attack allows an intruder to enter commands on the network. To carry out such an attack, the intruder needs to first pre-capture packets. By repeating these packets on the network, the attacker can confuse vehicle systems and even cause them to execute instructions based on erroneous data. To combat this, the proposed IDS detects which packets sent to the ECUs are benign and which are malicious.

The objective of the IDS is to learn the characteristics of Audio-Video Transport Protocol (AVTP) traffic, which is the protocol of the AVB set of standards that describes the transport of time-sensitive and prioritized traffic (i.e., Audio/Video data) [b6 2016], and distinguish whether a given packet is legitimate or malicious. An attack on the transmission of AVTP packets can corrupt video images being sent to the ECUs. This attack can be critical if it hits the driver assistance systems as the vehicle can make wrong decisions.

The proposed model requires real-time execution ideally in low-cost hardware, without needing a high-performance GPU. Furthermore, a comparison between a method known in the literature and the proposed IDS is performed. In a nutshell, the main contri-

butions of this work are:

1. The proposal of a real-time, XGBoost-based IDS to detect replay attacks on automotive Ethernet networks. The IDS can be deployed on low-cost hardware such as a Raspberry Pi while meeting Audio/Video timing requirements.
2. A comparative evaluation between the proposed IDS and a baseline CNN-based approach for intrusion detection in automotive Ethernet networks. This comparison shows that the proposed technique achieves an intrusion detection time of 620μs per sample, within the threshold value for real-time AVTP replay attack intrusion detection, which is 1000μs according to [Jeong et al. 2021b]. This value of 620μs is 56x faster than the intrusion detection time of the CNN-based model, with a loss in accuracy of just 1.72 percentage points in the test dataset.

The rest of this article is organized as follows. Section 2 presents the related works. Section 3 introduces the proposed IDS, describing the system design and the model training process. Section 4 shows the dataset used to create the intrusion detection model. Section 5 explains the experiments conducted. In Section 6 the results of the experiments are shown, in addition to an analysis of the results. Finally, Section 7 brings some ideas, and possible future works and concludes the paper.

2. Related Works

Currently, few works tackle the problem of detecting intrusion in automotive Ethernet networks. Most works in the literature on protecting automotive networks are focused on networks that use the Controller Area Network (CAN) protocol [Choi et al. 2018], [Markovitz and Wool 2017], [Kang and Kang 2016], [Freitas De Araujo-Filho et al. 2021]. This is mainly due to two reasons: the lack of an automotive Ethernet dataset and the fact that the use of Ethernet in automobiles is recent. With the increasing use of automotive Ethernet, solutions for this protocol are emerging.

The authors of [Jeong et al. 2021b] propose an IDS based on deep learning to detect replay attacks in automotive Ethernet networks. The authors claim that their work is the first to perform intrusion detection on automotive Ethernet, and because of that, it will be used as a baseline in this work. The proposed model is based on 2-dimensional convolutional neural networks (2D-CNN) and achieves 0.99% accuracy in identifying whether a packet arriving on the network is injected by an attacker or not. Deep Learning models require much overhead in terms of training and deployment. Due to their complexity, in general, high-cost and high-capacity hardware is required to perform real-time inferences on these models, i.e, even if the model is trained in a fog/cloud environment, inferences need to be made "inside" the vehicle, and a deep learning model, even after being trained, takes a long time to infer results. Because of this, the inference time required to run the model in real-time using a low-cost device makes it impractical to implement the IDS without a high-performance GPU, since deep neural networks have a high computational cost. For applications that need to be executed in real-time and in low-cost hardware, as is the case of the proposed scenario of our paper, deep learning-based models may be unfeasible.

The work in [Alkhatib et al. 2021] presents an IDS based on deep learning to prevent attacks on the SOME/IP protocol. The authors claim that their work is a pioneer in detecting intrusions in the Scalable service-Oriented Middle-warE over IP (SOME/IP)

protocol [AUTOSAR 2016], a type of middleware that can be used to improve intercommunication between multiple ECUs. However, the IDS proposed by the authors can only be used for offline detection. The intrusion can only be identified when the session between the client and the server ends.

The authors of [Alkhatib et al. 2022] propose an anomaly detection system in automotive Ethernet, more specifically in the AVTP protocol, based on the use of convolutional autoencoder (CAE). The dataset used by them is the same dataset used in [Jeong et al. 2021b]. The CAE algorithm used is an encoder and decoder with asymmetric CNN structures, anomalies in the AVTP packet flow can be detected by measuring the reconstruction error of an AVTP packet window. The authors claim to be the first to detect anomalies in automotive Ethernet-based networks. Their results show an accuracy of 0.94. However, their proposed model is only used for offline detection and cannot be used in real-time scenarios.

As a result, there are still few works that focus on proposing cybersecurity systems for automotive Ethernet networks. The existing works are not capable of performing intrusion detection in real-time, and those that have this capability require the use of high-performance GPUs, which may be costly for vehicles. To the best of the authors' knowledge, no other solution was found in the literature that aimed at real-time intrusion detection on automotive Ethernet networks and did not use a high-performance GPU. This makes the model presented in this paper the first in the literature to perform real-time intrusion detection on automotive Ethernet using low-cost hardware.

3. Proposed IDS

In this work, we propose a machine learning (ML) model based on XGBoost that detects intrusions on automotive Ethernet networks. The proposed IDS receives benign data while malicious data is purposely injected. After passing the switch, these packets are sent to the IDS, which initially uses these packets to generate features and then feeds these features into the machine learning model. The model aims at classifying whether the packet arrived is malicious or benign. The goal is to deploy the IDS on low-cost hardware that must be easily and cheaply added to a vehicle. Figure 1 illustrates our proposed system's architecture. The model proposed in this work can be fully reproduced by the methodology described in this section. As said before, the data used can be found in [Jeong et al. 2021a]

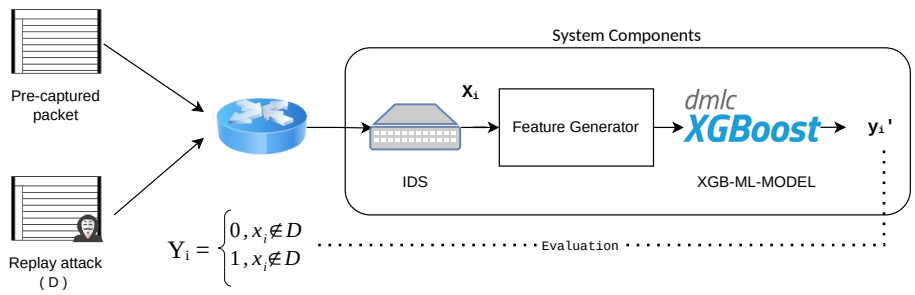

Figure 1. System design.

Replay Attacks Connected vehicles have a large attack surface and an adversary can enter the in-vehicle network in many ways, such as via OBD port, Wi-Fi, Bluetooth, mobile apps, USB ports, remote key, and others. Once inside the in-vehicle network, an attacker can perform a replay attack, which consists of sending and injecting valid frames into the network at the right time. Once the attacker is inside the in-vehicle network, the ECU cannot identify whether the source sending these frames is secure [Liu et al. 2017].

The proposed IDS aims to identify possible AVTP replay attacks. Such attacks can be dangerous as they have the ability to make ECUs make wrong decisions based on distorted information. Replay attacks can be simulated by repeatedly sending AVTP video frames. An example of the effects of a replay attack can be seen in Fig 2. Instead of receiving the original image, the ECU receives a distorted image.

Figure 2. On the left is the original frame that was supposed to be received by the ECU. On the right is the distorted image that is received when the system is under replay attack.

It is possible to divide the construction of the proposed intrusion detection system into the steps described below.

Data pre-processing. Initially, the processing and analysis of the data found in the dataset are carried out. This process is essential to understand the characteristics of the data and check if data processing is necessary before running the algorithm. Each AVTP packet in the dataset has a size of 438 bytes. The authors in [Jeong et al. 2021b] noted that in the first 58 bytes of packets, there are patterns that vary from packet to packet, making it possible to identify different patterns between malicious and benign packets. In the 59th-438th packets, no significant pattern is found in the byte shift. In this way, only the first 58 bytes of the packets generate features to input into the model.

Feature Generator. The Feature Generator process used in this work is based on the process proposed by [Jeong et al. 2021b]. After the data processing step, the feature generator is modeled. It will allow the machine learning model to find differences in the payload of packets after the insertion of malicious packets, which means that if there is a slight difference in the payload data, these packets are likely to be re-injected, and the model should detect this and classify them as malicious. The Feature Generator output is

used as input to the machine learning model. To start the Feature Generator, a window value is defined, where w is an integer such that $w \geq 4$. The objective is to aggregate the traffic that arrived in windows of size w, producing a two-dimensional vector. The authors in [Jeong et al. 2021b] noted that a window where $w = 44$ brings the best accuracy values for the model to be built so this window value will be used. A packet that arrives as input to the Feature Generator is represented by X_i, where $X_i = (X_{i,1}, X_{i,2}, ..., X_{i,(j-1)}, X_{i,58})$, X_i is an integer between 0 and 256 and $j = 58$, where j represents the j-th byte of the array. To generate the features, it is necessary to define a ΔX_i, representing the change of state of the i-th AVTP packet. In a benign packet these changes are smooth, while in a malicious packet they are more abrupt due to the insertion of distorted information. ΔX_i is represented in Eq. 1:

$$\begin{aligned} \Delta X_i &\equiv (X_i - X_{i-1}) \; mod \; 2^8 \\ &\equiv (X_{i,1}, X_{i,2}, ..., X_{i,(j-1)}, X_{i,58}) \; mod \; 2^8 \\ &\equiv (\Delta X_{i,1}, \Delta X_{i,2}, ..., \Delta X_{i,(j-1)}, X_{i,58}). \end{aligned} \tag{1}$$

In Fig 3 it is possible to observe the feature generation process. At the end of this step, a two-dimensional vector of size $w \times 2j$ is generated. In this step, the labels of each generated feature are also created. Label 0 is assigned to those packets considered benign, and label 1 to packets considered malicious. To label the packets that should be considered malicious in the training step, we define the following: a packet is considered malicious if at least one of its frames is injected by an attacker. These labeled data are used to train the model, and they must be able to identify if a packet is benign or malicious according to the payload characteristics. In this way, it is possible to train the proposed model using the data processed in this step.

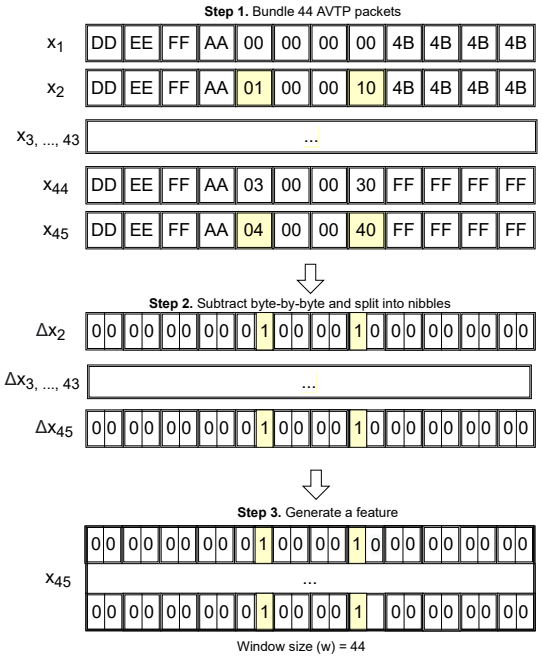

Figure 3. Feature Generator. Adapted from [Jeong et al. 2021b].

Training a ML model. The proposed machine learning model uses as a basis the XGBClassifier algorithm. The model was implemented using the XGBoost library available for python [Chen and Guestrin 2016]. The data $D_{indoors}$ has been used for training and validation and the data $D_{driving}$ has been used as a test set. The model to be built comprises an input which accepts 44×116-sized data. This training is performed using 5-fold cross-validation.

There are two types of XGBoost parameters: *Tree Booster parameters* and *Learning task parameters*. *Learning task parameters* set and evaluate the learning process of the booster from the given data. *Tree Booster parameters* control the performance of the tree booster since the model consists of a set of decision trees. The parameters to be defined in this process are of the *Tree Booster* type, which are:

- **learning_rate**. The rate at which our model learns patterns from data. Lower learning_rate leads to slower computation.
- **max_depth**. Maximum tree depth for base learners.
- **gamma**. Minimum loss reduction is required to make a further partition on a leaf node of the tree.
- **colsample_bytree**. The subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed.
- **min_child_weight**. The minimum sum of instance weight (hessian) needed in a child. If the leaf node has a minimum sum of instance weight (calculated by second-order partial derivative) lower than min_child_weight, the tree splitting stops. The larger min_child_weight is, the more conservative the algorithm will be.
- **reg_alpha**. L1 regularization term on weights. Increasing this value will make the model more conservative.
- **reg_lambda**. L2 regularization term on weights. Increasing this value will make the model more conservative.

More details can be found in [Chen and Guestrin 2016].

Initially, using the Hyperopt library [Bergstra et al. 2015], a hyperparameter tuning step was performed to find the best model. The parameters were varied according to the values in Table 1. The parameters selected after this step were used to train the model and can be seen in Table 2.

Table 1. XGB parameters and levels.

Parameter	Levels
max_depth	From 3 to 20
gamma	From 0 to 9
colsample_bytree	From 0.5 to 1
min_child_weight	From 0 to 10

Table 2. tuned XGB Model

Parameter	Levels
max_depth	6
gamma	0
colsample_bytree	1
min_child_weight	1
learning_rate	0.007
reg_alpha	4
reg_lambda	1

At the end of the training stage, the model to be used to detect malicious packets is obtained. In summary, the proposed model works in a few steps. In the first step, the sys-

tem bundles the packets being transmitted on the network into windows of size $w = 44$, selecting only the first 58 bytes of each frame - because it is in these bytes that the information allows to identify whether the packet is malicious. Then this frame is used as input to the Feature Generator. This step is essential because the Feature Generator allows identifying characteristics of the attacker's frames that are different from the original frames. Finally, the generated feature is given as input to the trained model that will identify whether that frame is malicious or not. This process repeats while AVTP packets are being sent over the network. Algorithm 1 shows the algorithm of our proposed IDS.

Algorithm 1 Proposed XGB-ML-Model

1. Use the pre-captured data to train the machine learning model called XGB_Model
2. **while** AVTP packets are received on the switch **do**:
3. create an array X, where X is the set of the first 58-bytes of 44 sequential AVTP frames
4. use array X as input to Feature Generator (Equation 1) and get feature
5. Use the feature as input to XGB_Model
6. **if** the frame is considered malicious **then**
7. discard frame
8. **end if**
9. **end while**

4. Dataset Description

The dataset used in this work contains real packet data in PCAP format and was provided by [Jeong et al. 2021a]. This dataset is made up of AVTP packets, where each packet is labeled as "benign" (if it is an actual packet) or "malicious" (if it is an injected packet by an attacker). These AVTP packets carry video frames, that is, each frame carries an image in its payload. These images are captured by cameras in the vehicles and sent to the ECUs, which can make decisions based on the data received. The authors responsible for the dataset initially built and collected attack-free stream AVTP data units (AVTP-DUs) packets in two scenarios: inside the lab and in a vehicle running on the road. In addition, video frames were inserted into the dataset to simulate a *replay attack*.

To simulate a replay attack, the authors of the dataset [Jeong et al. 2021b] insert a video frame containing 36 AVTP-DUs streaming packets while capturing the actual packets. This malicious video frame is available in [Jeong et al. 2021a], together with two datasets: the dataset built inside the lab called $D_{indoors}$ and the dataset built with the vehicle in motion, called $D_{driving}$.

5. Experiments Evaluation

The experiments were carried out on four machines. The first is provided by Google Collaboratory Pro with GPU NVIDIA Tesla P100 and has 26GB of RAM (NVIDIA Drive version is 460.32.03 and the CUDA version is 11.2.); the second uses a CPU Intel(R) Core(TM) i5-8250U @ 1.60GHz and has 16GB; the third uses a virtualized CPU ARM Neoverse-N1 with 1 Core and has 4GB of RAM; finally, the last one uses a Raspberry Pi 3 as a host with CPU ARM Cortex-A53 and has only 1GB of RAM (see Table 3). The data used to train the model were from the $D_{indoors}$ dataset, which has 446,372 AVTP

benign packets and 196,894 AVTP malicious packets. The $D_{driving}$ dataset was used for model testing, and it has 149,4257 benign packets and 376,236 malicious packets.

Table 3. Devices used in experiments.

Host	Processor	Available RAM
Google Collaboratory Pro	(GPU) NVIDIA Tesla P100	26GB
Laptop X510URR	(CPU) Intel(R) Core(TM) i5-8250U	16GB
Oracle VM	(CPU) ARM Neoverse-N1 (Single Core)	4GB
Raspberry PI 3b	(CPU) ARM Cortex-A53	1GB

The fourth host in Table 3 was used to estimate the performance of the model on low-cost hardware. This means that good performance on the Raspberry Pi 3 host indicates that the model can be used for real-time intrusion detection on other inexpensive devices, such as the new Raspberry Pi 4. To get an idea, the Raspberry Pi 4 has a Quad-core Cortex-A72 (ARM v8) CPU that has higher specs than the processor used in our tests, with the same architecture.

5.1. Model Train

The training process was done using the dataset data $D_{indoors}$ for training and validation. The data $D_{driving}$ was used for the test. The model is trained using the parameters found in the hyperparameter optimization step and can be seen in Table 2. This training process is performed using 5-fold cross-validation.

5.2. Evaluation metrics

The evaluation metrics used are Accuracy, Precision, Recall and F1-Score. These metrics are calculated as follows:

$$\text{Accuracy} = (TP + TN)/(TP + TN + FP + FN);$$

$$\text{Precision} = (TP)/(TP + FP);$$

$$\text{Recall} = (TP)/(TP + FN);$$

$$\text{F1} - \text{Score} = 2 \times \text{Precision} \times \text{Recall}/(\text{Precision} + \text{Recall}),$$

where TP is the number of True Positive classifications, TN is True Negatives, FP is False Positives, and FN is False Negatives. F1-Score is the harmonic mean of precision and recall. In addition, the metrics Area under the ROC curve (ROC AUC) and Average inference time per sample are also used.

ROC AUC is a standard metric used in binary classifiers and shows changes in the true-positive rate and false-positive rates. The inference time is a critical metric since the IDS must be executed in real-time: if an attack occurs, it must be identified before the packet reaches the ECU and causes errors and anomalous behavior.

6. Results and Analysis

As mentioned before, the model proposed in [Jeong et al. 2021b] is used as a baseline. Here, this model is identified as "2D-CNN-Model". The model proposed in this work is identified as "XGB-ML-Model". It is essential to highlight that in order to evaluate both models under the same conditions and on the same hosts, the 2D-CNN-Model model was faithfully implemented and based on the authors' specifications in [Jeong et al. 2021b].

Tables 4 and 5 show the average results of the evaluation metrics for both models using 5-fold cross-validation. The individual values can be found in a csv file [1]. Table 4 shows the results obtained for the training set. The main metric used is the F1-score since the data is slightly imbalanced, and the F1-score is a better metric when there are imbalanced classes. For the baseline 2D-CNN-Model, F1-scores is **0.9927**. With XGB-ML-Model the F1-core value found is **0.9993**. This means that the models can classify almost all AVTP packets correctly.

The results for the test set $D_{driving}$ can be seen in the Table 5. For XGB-ML-Model, the value of F1-score is **0.9805**, slightly lower than the average value found in the 2D-CNN model. Also noteworthy is the better performance in the Precision metric of the XGB-ML model in relation to the 2D-CNN model, although it is worse when comparing the other metrics.

Table 4. Classification results in training dataset.

	Accuracy	Precision	Recall	F1-score	AUC
2D-CNN-Model	**0.9955**	0.9913	**0.9940**	**0.9927**	**0.9997**
XGB-ML-Model	0.9847	**0.9970**	0.9823	0.9896	0.9983

Table 5. Classification results in test dataset.

	Accuracy	Precision	Recall	F1-score	AUC
2D-CNN-Model	**0.9919**	0.9637	**0.9979**	**0.9804**	**0.9990**
XGB-ML-Model	0.9747	**0.9727**	0.9357	0.9538	0.9805

In Table 6, the average values of inference time are obtained. The values discussed here refer only to the average time each model takes to classify the packet on each processor, that is, the time it takes from the moment the method is invoked until the moment the packet is classified. It is possible to observe the difference in performance between the models, especially if we consider the model running on a CPU. The inference time of the 2D-CNN model on CPU i5-8250U is about 10x slower than that for XGB-ML on the same platform, 27x slower when compared to that for XGB-ML on the CPU ARM Neoverse N1, and 56x slower than that for XGB-ML when deployed on a Raspberry Pi 3. Unlike 2D-CNN, the XGB-ML model does not benefit from running on GPU while maintaining excellent performance. It is important to keep in mind the threshold value for real-time AVTP replay attacks intrusion detection, which is **1000** μs according to [Jeong et al. 2021b].

[1] https://raw.githubusercontent.com/prximenes/ML-based-ids-paper-results/main/5fold.csv

Table 6. Average inference time per sample.

Model	Processor	Time (µs/sample)
2D-CNN-Model	(GPU) NVIDIA Tesla P100	83
	(CPU) Intel(R) Core(TM) i5-8250U	980
	(CPU) ARM Neoverse-N1 (Single Core)	5246
	(CPU) ARM Cortex-A53	**35,000**
XGB-ML Model	(GPU) NVIDIA Tesla P100	96
	(CPU) Intel(R) Core(TM) i5-8250U	92
	(CPU) ARM Neoverse-N1 (Single Core)	193
	(CPU) ARM Cortex-A53	**620**

Discussion. Cost, performance, and safety are concerns that need to be considered when one develops any system for automobiles. High cost or even high weight solutions may be unfeasible to be introduced in vehicles. Recently, there has been a discussion about moving to increasingly vehicle-centric architectures in order to meet cost, performance, and safety requirements [Bandur et al. 2021].

The solution used as a baseline and proposed by [Jeong et al. 2021b] achieves high accuracy in both sets where it was tested. However, when we take into account the average inference time, a fundamental metric to assess whether the solution is capable of running in real-time, we can observe that the performance of the solution suffers a considerable drop when it is executed in CPUs and even in lower performance power GPUs (see [Jeong et al. 2021b]). The solution proposed in this work, on the other hand, manages to maintain the high accuracy of the Deep Learning model proposed in [Jeong et al. 2021b], showing slightly lower performance in the Recall and F1-score metrics, but even surpassing the baseline model considering the Precision metric. Despite this, it is essential to note that the proposed model can run in real-time on a Raspberry Pi 3, without needing a high-performance GPU. This must be considered, as it is hardly possible to have a high-performance GPU in a vehicle dedicated to executing an intrusion detection algorithm. At the same time, the availability, cost, and even weight of a Raspberry Pi like the one used to test the models in this work show that it can be used without difficulty.

The model proposed in this work is below the threshold of 1000 μs on all platforms used, especially on the Raspberry Pi 3 and ARM-based Neoverse-N1 CPU. It is also important to point out weight and power consumption concerns. A high-performance GPU like the one used in the experiments weighs about 1.5Kg and has a maximum power consumption of 250W, while a Raspberry Pi 3 weighs 0.46g and consumes about 15W. Thus, the cost-benefit of using the XGB-ML model based on decision trees proposed in this work is evident. The IDS proposed based on XGB-ML is able to differentiate the transmission of legitimate AVTP packets and the transmission of packets through replay attacks. Also, it can do this in real-time, using an inexpensive device like a Raspberry Pi 3. Manufacturers can choose to reduce cost, weight, and power consumption and keep a model running on a cheap, lightweight device, even if it means a slight loss of accuracy. The intrusion detection system should be just one of many other security redundancies running on the system and network, and because of that, it should not cost more or take

up more space in the project than necessary.

7. Conclusion and Future Works

In this work, a machine learning-based real-time intrusion detection system to detect AVTP replay attacks in automotive Ethernet was proposed. The IDS can be implemented in low-cost hardware and without the need for a high-performance GPU. In addition, the proposed approach was compared with another known in the literature to detect intrusion in automotive Ethernet networks and both were discussed and analyzed. The first approach, based on 2D-convolution neural networks, was proposed by [Jeong et al. 2021b] and managed to achieve excellent results in terms of precision and accuracy. However, it presents very high inference times when executed without using high-performance GPUs, which can affect its ability to detect AVTP replay attacks in real-time.

The approach proposed in this paper is based on decision trees through an XGBoost classifier, and despite having slightly lower performance in terms of precision and accuracy than the 2D-CNN model, it can perform inferences in real-time using an inexpensive Raspberry Pi, without the need for a high-performance GPU. The paper illustrates how algorithms simpler than deep neural networks can be used as an alternative to creating simple models that run in real-time without the need for a high-performance GPU. The challenges and limitations of each model were also discussed.

In future works, it is important to find a way to keep the classification metrics of the model at the highest possible levels, while keeping the inference times compatible with running the IDS in real-time. A way to achieve this goal is to train a model based on deep learning that can perform real-time inferences even on limited or low-cost hardware. Examples of neural networks that could be investigated are depth separable convolutions, convolution networks that utilize pointwise group convolution and channel shuffle, and light-weight vision transformers. These networks were proposed to be "Mobile-Friendly" and can be adapted to solve the problem seen in this work in real-time on devices with low-cost hardware. In addition to these networks, 1D-Conv networks have recently been used and achieved state-of-the-art performance levels in various applications and can be investigated.

Acknowledgement
This work was supported by Fundação de Amparo à Ciência e Tecnologia de Pernambuco (FACEPE), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), and Fonds de recherche du Québec B2X Scholarship. We also thank the authors of [Jeong et al. 2021b] for their support in reproducing their work.

References

(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.

Alkhatib, N., Ghauch, H., and 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), pages 0954–0962.

Alkhatib, N., Mushtaq, M., Ghauch, H., and Danger, J.-L. (2022). AVTPnet: Convolutional Autoencoder for AVTP anomaly detection in Automotive Ethernet Networks. *arXiv preprint arXiv:2202.00045*.

AUTOSAR (2016). SOME/IP protocol specification. [online] Available: `https://www.autosar.org/fileadmin/user_upload/standards/foundation/1-0/AUTOSAR_PRS_SOMEIPProtocol.pdf`.

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.

Bello, L. L. (2011). The Case for Ethernet in Automotive Communications. *SIGBED Rev.*, 8(4):7–15.

Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., and Cox, D. D. (2015). Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008.

Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

Choi, W., Joo, K., Jo, H. J., Park, M. C., and Lee, D. H. (2018). Voltageids: Low-level communication characteristics for automotive intrusion detection system. *IEEE Transactions on Information Forensics and Security*, 13(8):2114–2129.

El-Rewini, Z., Sadatsharan, K., Selvaraj, D. F., Plathottam, S. J., and Ranganathan, P. (2020). Cybersecurity challenges in vehicular communications. *Vehicular Communications*, 23:100214.

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*, 9:166855–166869.

Jeong, S., Jeon, B., Chung, B., and Kim, H. K. (2021a). Automotive Ethernet intrusion dataset. Available at `https://dx.doi.org/10.21227/1yr3-q009`.

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

Kang, M.-J. and Kang, J.-W. (2016). A novel intrusion detection method using deep neural network for in-vehicle network security. In *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE.

Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., and Savage, S. (2010). Experimental Security Analysis of a Modern Automobile. In *2010 IEEE Symposium on Security and Privacy*, pages 447–462.

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.

Markovitz, M. and Wool, A. (2017). Field classification, modeling and anomaly detection in unknown CAN bus networks. *Vehicular Communications*, 9:43–52.

Matheus, K. and Königseder, T. (2021). *Automotive Ethernet*. Cambridge University Press, 3 edition.

Porter, D. (2018). 100BASE-T1 Ethernet: the evolution of automotive networking. *Texas Instruments, Techn. Ber*.

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.