Learning-based Intrusion Detection System for On-Board Vehicle Communication

Tobia Fiorese [1,2] and Pietro Montino [2]

[1] *University of Padova, Italy*
[2] *Bluewind, Via della Borsa, 16A Castelfranco Veneto, 31033 (TV), Italy*

Abstract
This paper presents the development of an Intrusion Detection System (IDS) to be deployed on a CAN bus. Since the introduction of many external interfaces in modern vehicles exposes users to the risk of cyber-attacks, the need of focus on security is concrete. An IDS is a component that can detect anomalies in the behavior of the system where it is deployed. The proposed machine learning based solution is composed of two parts. The first includes a supervised trained neural network that is able to distinguish among different known attacks. The second includes a discriminator that has been trained exploiting the Generative Adversarial Network (GAN) paradigm, to distinguish among the attack-free situation and an anomalous situation. It will be demonstrated how the supervised training can achieve state of the art performance in classification and how the unsupervised training can guarantee a certain level of security even without the necessity of feeding labeled data to the network.

Keywords
GAN Training, IDS, CAN Bus

1. Introduction

A connected car is a car capable of communicating bidirectionally with systems located outside of itself or with internal devices. Many forms of external interfaces were added to modern vehicles in recent years in order to enable such communication, unlocking a broad spectrum of new comforts for drivers. Among them are the collection of real-time traffic information, accident emergency services, access to status and operating conditions of the car.

On the one hand, all of the above-mentioned features enrich the vehicle’s user experience and enhance higher transportation network security and efficiency standards. On the other hand, all these communication interfaces expose the vehicle internal networks to remote attacks, thus increasing the vehicles’ cyber vulnerability.

In the future, each car will constantly communicate with the surrounding environment made of other cars, pedestrians and road signs giving rise to what is called the Vehicle to Everything (V2X) paradigm [1]. This will dramatically increase the risk of potential attacks that could exploit the greater connectivity routes available.

In a modern vehicle there are many kinds of networks: the CAN bus is the de facto standard for safety critical applications. The CAN bus connects hundreds of Electronic Control Units (ECUs) and controls most, if not all, of the electro-mechanical actuated systems: from brakes to lights, from airbags to transmission. Being designed in the eighties, CAN bus does not embed security features. In today’s connected world, finding ways to prevent cyber-attacks that could damage on-board systems and put the vehicle and the passengers at risk, has become of pivotal importance.

Side by side with the growing awareness of the vulnerabilities in CAN bus security, and the consequences that a cyber-attack on a vehicle could entail, some technical countermeasures have already been adopted. Since the 2000s, preventive solutions have been explored such as the use of

ITASEC21, May 2021
EMAIL: fioresetobia@gmail.com (A. 1); pietro.montino@gmail.com (A. 2)
ORCID: 0000-0002-0151-5597 (A. 1); 000-0001-9715-9998 (A. 2)

sub-networks along with the addition of gateways and firewalls that block messages trying to cross subnets. Authentication protocols [2] and encryption methods [3] have been deployed as well, thus increasing safety levels of CAN buses. Other countermeasures include Intrusion Detection Systems (IDS), that try to identify anomalies on the traffic on a specific network by reading all the messages in transit.

This paper presents the architecture and the safety performances of a two-step CAN bus IDS trained with machine learning techniques. This approach allows to add a measure of safety, even for on-board electronics architectures already consolidated, without too much interference with pre-existent hardware and software.

2. Related Works

In the automotive field, numerous anomaly detection methods have been proposed. Most of these solutions try to identify two categories of attacks: those based on the frequency of messages, such as insertion or deletion of packets, and those that manipulate the payload.

In [4] attacks are detected by checking the validity of the single frames with formal rules, and by defining a certain number of forbidden message sequences by exploiting the correlation between the target frame and contextual information regarding previously transmitted messages. A different rule based on transition matrices was illustrated in [5].

Another approach is presented in [6], where anomalies are detected by looking at the inter frame arrival times and at the frequencies of particular messages. This method can identify deviation from normal traffic targeting particular ECUs, and is pretty accurate, but frequency alone cannot be relied upon to discriminate an attack from an irregular noise situation in the CAN network.

It is also possible to undertake a statistical approach, as it was done in [7], where a rolling window moves over a sequence of messages, focusing the attention on a portion of data in transit and computing statistical measures as the standard deviation on offsets and time intervals between messages.

Also, many learning-based approaches were tested. Some of them are based on the time series of payload values, and make extensive use of recurrent neural networks, in particular LSTM models [8]. However, this approach takes advantage of knowledge of payload semantics, which in most cases is unknown and proprietary. Another way to detect attacks in the network is to use a compound classifier, fusing a one-class SVM for each ID, to obtain an overall anomaly score [9]. Some other interesting approaches use a rolling window over the messages' sequence and then use a CNN to classify the matrices obtained from a pre-processing on the portion of messages focused by the window [10]. This method proved to be very accurate in detecting known attacks and has been further improved by adding an additional filter on packet traffic consisting of a CNN trained with the GAN technique [11].

3. CAN Bus

CAN [12] is a network with a bus topology, which means that all devices on the network are connected to a single line, or bus.

At the transfer layer, the information is sent over the CAN bus through messages with a fixed format and of limited length. Each frame is made up of various fields, the most significant of which are:

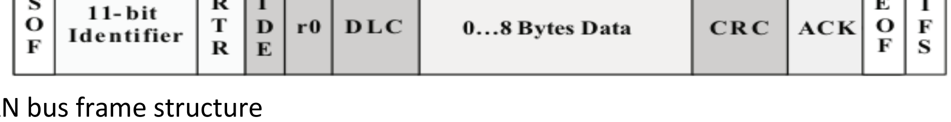

Figure 1: CAN bus frame structure

- **Identifier**, an 11-bits identifier that also defines the priority of the message. The lower the binary value, the higher the priority. Can be extended to 29 bits.
- **Data**, 64-bit field whose semantics are generally proprietary and specific to each ECU.

A characteristic worth noticing is that there are no source and destination fields in a CAN packet: message routing is regulated by the ID field. A CAN packet is broadcasted to the entire network, and only some ECUs will process the packet based on the ID received.

4. Attacks

The first neural network of the presented IDS (ANN1) is trained in a supervised manner with a dataset of different categories of known attacks. The goal of ANN1 is to discriminate between known attacks and the normal condition by inferring over a sequence of data packets, in real time. The datasets used for training are made of real CAN bus messages [11], [7]. They consist of logs of traffic directly picked from the CAN bus through the OBD-II port of commercial vehicles where attacks were carried out connecting a Raspberry Pi3 board to the bus and a laptop to the board. The following paragraphs describe the types of known attacks that get detected by ANN1.

4.1. Denial of Service (DoS)

An attacker can inject high priority messages in a short cycle on the bus. DoS attack messages aim at occupying the bus using the theoretically highest priority identifier, namely 0x000. Since all nodes share a single bus, increasing occupancy of the bus can produce latencies of other messages and cause threats regarding availability with no response to driver’s commands

4.2. Fuzzy

An attacker can inject messages with randomly spoofed CAN ID, either with arbitrary data or with spoofed data values. All these messages are functional and structurally correct, but they can cause unintended vehicle behaviors. An attacker can passively observe in-vehicle traffic and select target identifiers to produce unexpected behaviors. Unlike the DoS attack, Fuzzy is more specific and aims at paralyzing a particular function of the vehicle.

4.3. Impersonation

An attacker can manage to stop message transmission from a target node and can plant/manipulate an impersonating node that will take its role. If a victim node stops transmitting, all messages sent by the targeted node will be removed from the bus.

5. Pre-Processing

The two datasets have been divided into windows of traffic logged from the bus. Each of these windows contain a number of packets, that during tests varied from a minimum of 8 to a maximum of 128. The most informative fields in CAN packets are the ID and the payload. To enhance the system flexibility, semantic comprehension of payload data was not accounted.

Only sequences of IDs were used to train the models. However, in log files IDs are formatted as hexadecimals, therefore they are not suited to be fed to a neural network directly. Windows of IDs have been converted into grayscale images with the two encodings pictured below.

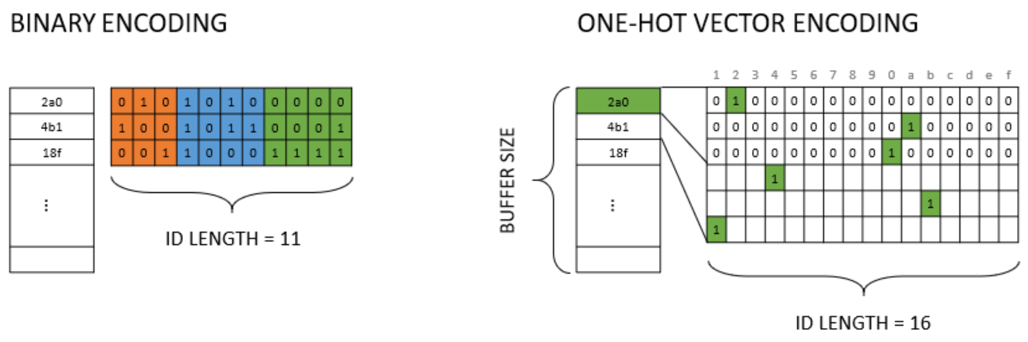

Figure 2: Encoding process of datasets

6. Supervised Training

In this chapter the different neural network architectures will be illustrated. Each of these architectures have been mantained small enough to fit memory constrained hardware. Two strategies ensured high detection accuracies: using RNNs, being the relationship of IDs sequences temporal rather than spatial, and using data fusion via the exploitation of other data available from the datasets. Each model proposed have been tested on different window sizes ranging from 8 to 128 CAN messages. All models have been trained using the same optimizer and the same hyperparameters.

6.1. CNN Model

The CNN model consists of 3 convolutional layers and a fully connected layer with 32 units which precedes the last layer that outputs four probability values, one for each class of attacks. Each convolutional layer has 3x3 filters. The number of filters increases while going deeper in the network. Each convolutional layer is followed by a BatchNormalization, a ReLU activation function and a Dropout layer (at training time only).

6.2. RNN Model

The RNN model consists of 32 LSTM cells followed by two fully connected layers with respectively 64 and 32 units. Finally, the output layer is the same as the one in the CNN model. In this way, in respect to CNNs complexity of models is consistently reduced.

6.3. Data Fusion

Looking at time intervals between subsequent messages enables the detection of a set of known attacks [7]. The sequence of timestamps available in the datasets was divided into windows. Each window was normalized by subtracting its first value from each timestamp and then dividing by the mean time elapsed in the transmission of a normal window, to remove outliers.

The data fusion models were built as the concatenation of two models, one taking as input the sequences of IDs and the other taking as input the sequences of timestamps. The first model has the same structure as defined in the previous section. The second has the same architecture but uses one-dimensional filters. After the convolutional layers, each model output is flattened and concatenated.

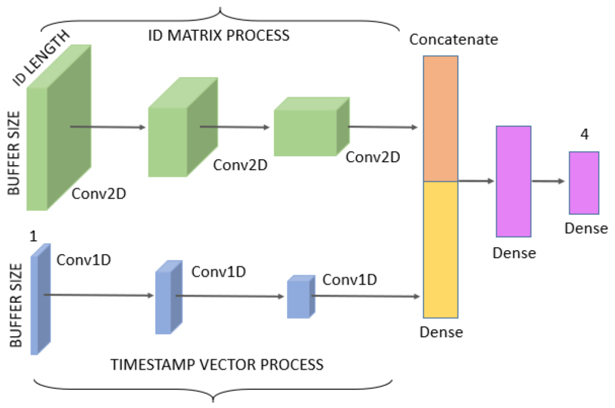

Figure 3: Architecture of data-fusion CNN model

As expected, the number of trainable parameters grows compared to above models. However, at the expense of a limited increase in model complexity, the performance in classifying attacks is considerably enhanced, in particular for convolutional models and bigger windows.

6.4. One-Hot Vector Encoding

One-hot vector encoding proved to offer better results in the classification of images extracted from CAN bus traffic [11].

The CNN model tested with OHV encoded data is composed of three convolutional layers, with respectively 4, 8 and 16 channels. Each layer has a filter with a 3x5 kernel that is moved over the image with a stride of 1x2. At the end, between the flattened matrix and the output layer there is a dense layer composed of 32 neurons. The use of a rectangular stride allowed for a consistent reduction of the number of parameters.

The RNN model instead maintains the same structure presented in 6.2, with the only difference in the input size.

7. GAN Training

A drawback of the supervised approach to machine learning in anomaly detection is that it relies on known attacks. Slight variations to attacks forming the dataset will increase the possibility to be confused with normal situations. Approaches based on distinguishing the normal behavior from anomalies, based only on attack-free data, could have a high false positive rate. It is possible though to train a network to generate data similar to the ones of the given dataset, and at the same time train a second network to distinguish among generated data and real data. This mechanism is called Generative Adversarial Network training. This approach results in a reliable discriminator that could enable protection against unknown attacks and also helps reduce the false positive samples yielded by a supervised model. In the next sections architectures and setups of the second part of the proposed system (ANN2) will be illustrated.

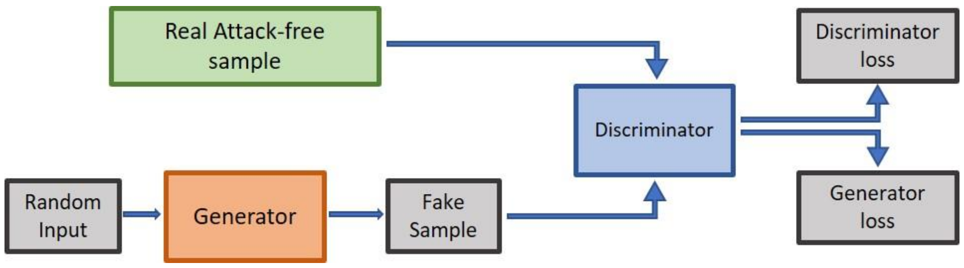

Figure 4: GAN training flow diagram

7.1. CNN Discriminator

The CNN discriminator is composed of 3 convolutional layers, each of them with a 3x3 kernel and respectively 128, 64 and 32 channels, and a single neuron as output. Each layer is followed by a Leaky ReLU function and a Dropout layer.

The associated generator takes a random vector of 256 values as the input of a dense layer with 13*5*8 units, that is reshaped in a three-dimensional object. The network is then composed of 3 Conv2DTranspose layers. The first two consist of rectangular filters of dimension 5x3, moved with a stride of 2x1. They have 4 and 3 filters respectively. The last layer has only one channel and uses a 4x3 kernel moved in 1x1 strides. Each layer is followed by a ReLU function and a Dropout layer. The last layer squeezes the values of the generated image in the range [-1, 1] applying a 'tanh' activation function.

7.2. DNN Discriminator

The DNN discriminator is composed of 2 dense layers, with bi-dimensional input and respectively 96 and 48 units, and a single neuron as output. Each layer is followed by a Leaky ReLU function and a Dropout layer.

The associated generator takes a random vector of 256 values as the input of a dense layer with 1*12*256 units, that is then reshaped in a three-dimensional object. The network is then composed of four Conv2DTranspose layers. Each of them moves its filters along a stride of 2x2. The first two layers have 3x3-sized filters while the last two have 5x5. Each layer is followed by a ReLU function and a Dropout layer. The last layer squeezes the values of the generated image in the range [-1, 1] applying a 'tanh' activation function.

7.3. Training Setup

Each architecture of ANN2 has been tested with three different setups:

- **DCGAN**: the last neuron of the discriminator incorporates a sigmoid activation function. The loss function used is the Minimax loss and the optimizer used is Adam with a learning rate of 1e-3. BatchNormalization is added after each layer of the two networks.
- **WGAN**: the last neuron of the discriminator can output any real value, however the weights of each layer in the discriminator are clipped in the range [-0.01, 0.01] after each iteration. The loss function used is the Wasserstein loss and the optimizer used is RMSprop with a learning rate of 5e-5. The discriminator is trained for 5 times the iterations of the generator.
- **WGAN-GP**: the last neuron of the discriminator can output any real value, however a penalization factor (Gradient Penalty) is added to the loss after each iteration. The loss function used is the Wasserstein loss and the optimizer used is Adam with a learning rate of 2e-5, beta_1=0.5 and beta_2=0.9. The discriminator is trained for 5 times the iterations of the generator. Gradient penalty factor λ has been fixed to 10.

DCGAN	WGAN	WGAN-GP
D Loss: $E_x[log(D(x))] + E_z\left[log\left(1 - D(G(z))\right)\right]$	D Loss: $D(x) - D(G(z))$	D Loss: $D(x) - D(G(z)) + \lambda(\lVert \nabla_{\hat{x}}(D(\hat{x})) \rVert_2 - 1)^2$
G Loss: $log\left(D(G(z))\right)$	G Loss: $-D(G(z))$	G Loss: $-D(G(z))$

$D = discriminator\ function,\quad G = generator\ function,\quad x = real\ sample,\quad z = random\ noise\ vector$

Figure 5: Loss functions of GAN training setups

8. Results

8.1. Supervised Training

In **Figure 6**, average accuracies of architectures presented in Supervised Training, over OTIDS dataset, can be compared over different window sizes. Accuracy is calculated as the sum of true positives and true negatives over the sum of all the samples of each class in the test dataset. The AUC score is also reported in **Figure 6**. Detailed measurements on individual classes and on GIDS dataset are reported in Appendix A.

Analysing the results, a trend can be extrapolated from each test, that shows how the accuracy in detecting a particular situation is almost everywhere increasing with bigger windows. This is possibly due to the fact that the model has more data to relies on to predict the various classes. Instead, with smaller windows, the model is biased to predict more a single class with respect to the others. Following the same principle of adding data to get a more reliable detection, data fusion models prove to be more accurate.

As it is clear, RNN models yields better results than CNN on almost all the different attacks and window sizes. Also, the limited complexity of the models let this solution be the best choice to be deployed on an ECU. Using the 64-message long window on the combined RNN, the average accuracy achieved is 99.77%.

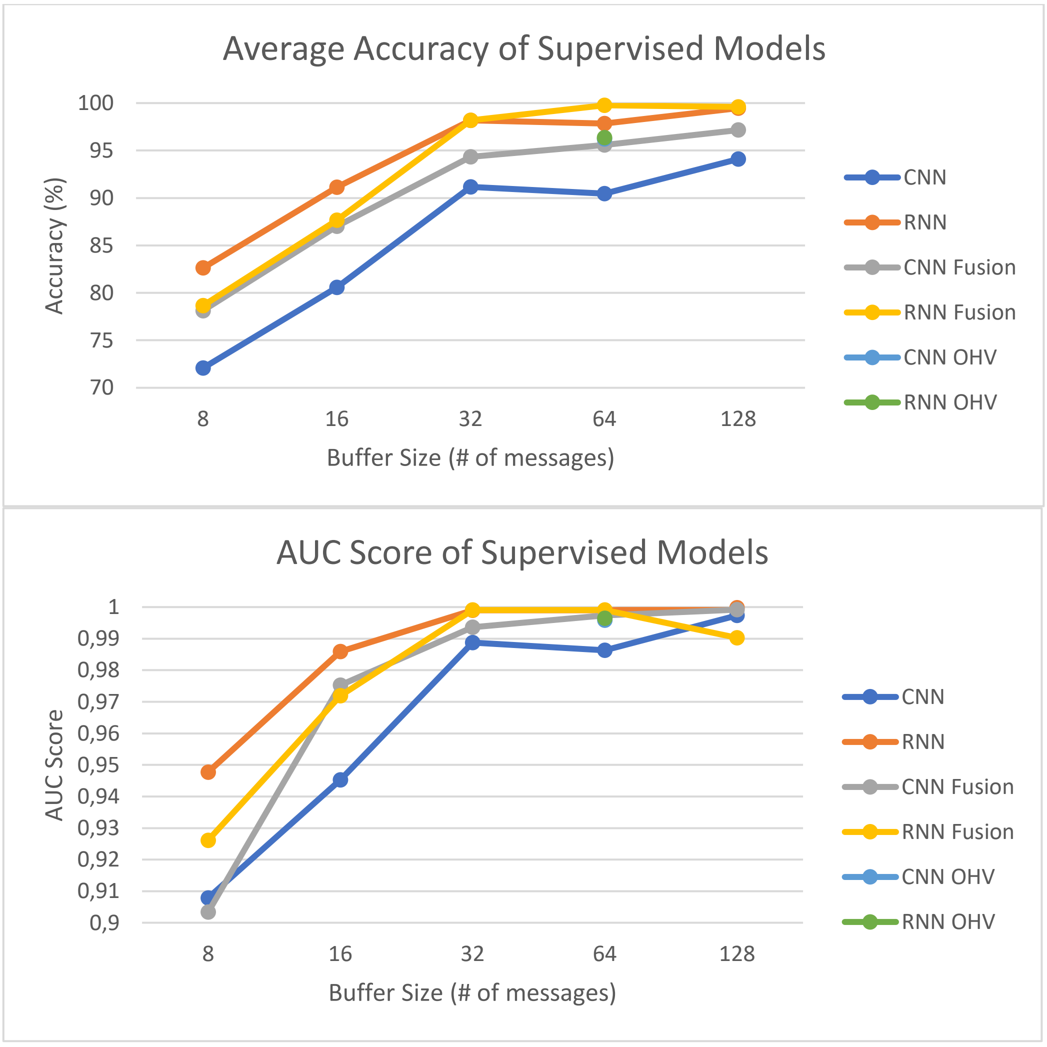

Figure 6: Average accuracy and AUC score of supervised models on OTIDS dataset

Looking now at the different encodings, fixing the window to 64 messages, there is a positive aspect in using OHV for CNN models. However, for RNN models there is a drop in accuracy of detecting Impersonation attack. It is also worth noticing that inputs for OHV models are almost four times bigger. This results in an increasing complexity of models.

8.2. GAN Training

Evaluation have been done comparing the output of the discriminator with a threshold, that can be roughly evaluated from the mean outputs on test set. Assuming that the outputs of the discriminator can be modelled as a normal distribution, a threshold have been fixed in a way that 99% of attack-free data can be correctly classified. This choice would be beneficial in a possible use case, where it is desirable a low false negative rate.

Confirming the tests on [11], CNN discriminator proved to fail in distinguishing attacks from the normal situation, However, one of the main purposes of GANs has been achieved. Looking at the generated data they seem pretty similar to the real ones. Problems in convergence were encountered, as the loss functions reached a stalemate after few iterations. In this way the discriminator had only the chance to see a small amount of different samples, which do not allow it to differentiate the output on the various attacks. This problem can be summarized as Mode Collapse.

DNN discriminator making use of Minimax loss suffer of convergence problems. In OTIDS dataset the output is similar to the one presented above, where known attacks cannot be discriminated. Looking at the distribution of outputs of the model at the end of the training, it can be easily seen that the intervals, in which each class falls in, overlap. Setting the threshold in a way that most of attack free data can be correctly classified lead to poor results in classifying attacks properly.

DNN discriminator with WGAN setup managed to reach a proper classification for two out of three attacks in OTIDS dataset. The only one that is almost always misclassified as an attack-free case is the Impersonation attack. In this setup, an important parameter to tune is the clipping range. Increasing it can spread out the outputs, but beyond a certain limit the training becomes very unstable.

DNN discriminator with GP-WGAN setup achieved the best results. In OTIDS dataset a perfect separation between Fuzzy and DoS attacks from Normal data have been reached, However, Impersonation still cannot be detected. Thus, it was possible to achieve a minimum level of protection against specific types of attack, starting only from data concerning an attack-free situation.

Table 1
Accuracy of GAN and Combined Method prediction on OTIDS dataset

	DCGAN	WGAN	WGAN-GP	CNN OHV + DCGAN	CNN OHV + WGAN	CNN OHV + WGAN-GP
No attack	96,66	98,93	98,15	99,8	100	99,64
DoS attack	5,26	81,79	100	99,93	100	100
Fuzzy attack	4,05	54,62	100	99,86	99,93	100
Impersonation attack	2,28	0,57	2,63	95,94	95,73	96,02

9. Combined Detection

The system used for attack detection is composed of ANN1 and ANN2. When the prediction of ANN1 is not correct the input is passed to ANN2 that gives its response. Even if the performance of discriminators themselves were not excellent, the combination of models improved them. Here below results obtain on the OHV encoded dataset are reported. In all of them the supervised model used was the CNN one, that proved to be the weakest, that is used as a binary classificator. Only results for DNN discriminators are reported, as CNN ones proved to be useless and so they cannot bring any improvement.

10.Conclusion and Further Works

In this paper, we showed how we could reach state of the art performance on known attacks classification using models with limited complexity. Furthermore, we showed how GAN training can help expand the domain of training by going beyond known attacks, improving the results achieved with supervised training techniques alone. Even more, we showed how the single use of the GAN paradigm can guarantee a certain level of protection against cyber-attacks even if the model used has never experienced any of them. That is, only data taken from an attack-free situation directly collected from the system we want to protect are necessary to achieve a minimum amount of security.

Of course, further research could be carried out on the basis of the methods proposed. Data fusion can be extended with other data collected directly from the CAN bus. An example can be the clock skew, that has already been used in other solutions, and that is more robust than timestamps. Furthermore, from the supervised point of view other combinations of models, encoding and different pre-processing techniques could be tested.

From the GAN perspective, many other models have been presented in recent years, like the StyleGAN or the CycleGAN. Furthermore, other techniques that haven't been explored could be added to the presented models, for example the spectral normalization of models' weights, in order to check for faster convergence during GAN training.

However, to understand the degree of security that our system could provide, it would be a great improvement to have access to more data. The effective deployment of the model on top of a real CAN bus could result in a different behaviour.

We expect that Generative Adversarial Networks will play an always greater role in the design of security and safety systems in the future and that the field of GAN will be a field of fervent academic and industrial research.

11.References

[1] J. Wang, Y. Shao, Y. Ge e R. Yu, A Survey of Vehicle to Everything (V2X) Testing, Sensors, vol. 19, p. 334, 2019. doi: 10.3390/s19020334.

[2] A. V. Herrewege, D. Singelee and I. Verbauwhede, CANAuth - A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus, in: Proceedings of the ECRYPT Workshop on Lightweight Cryptography 2011, Louvain-la-Neuve, Belgium, pp. 229-235.

[3] A. S. Siddiqui, Y. Gui, J. Plusquellic e F. Saqib, Secure communication over CANBus, in: Proceedings of the IEEE 60th International Midwest Symposium on Circuits and Systems, MWSCAS 2017, Boston, MA, USA, August 6-9, 2017, pp. 1264-1267. doi: 10.1109/MWSCAS.2017.8053160.

[4] I. Studnia, E. Alata, V. Nicomette, M. Kaâniche e Y. Laarouchi, A language-based intrusion detection approach for automotive embedded networks, Int. J. Embed. Syst., vol. 10, p. 1–12, 2018. doi: 10.1504/IJES.2018.10010488.

[5] M. Marchetti e D. Stabili, Anomaly detection of CAN bus messages through analysis of ID sequences, in: Procedings of the IEEE Intelligent Vehicles Symposium, IV 2017, Los Angeles, CA, USA, June 11-14, 2017, pp. 1577-1583. doi: 10.1109/IVS.2017.7995934.

[6] A. Taylor, N. Japkowicz e S. Leblanc, Frequency-based anomaly detection for the automotive CAN bus, in: Proceedings of the 2015 World Congress on Industrial Control Systems Security, WCICSS 2015, London, United Kingdom, December 14-16, 2015, pp. 45-49. doi: 10.1109/WCICSS.2015.7420322.

[7] H. Lee, S. H. Jeong e H. K. Kim, OTIDS: A Novel Intrusion Detection System for In-vehicle Network by Using Remote Frame, in: Proceedings of the 15th Annual Conference on Privacy, Security and Trust, PST 2017, Calgary, AB, Canada, August 28-30, 2017, pp. 57-66. doi: 10.1109/PST.2017.00017.

[8] A. Taylor, S. Leblanc e N. Japkowicz, Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks, in: Proceedings of the 2016 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016, Montreal, QC, Canada, October 17-19, 2016, pp. 130-139. doi: 10.1109/DSAA.2016.20.

[9] A. Tomlinson, J. W. Bryans e S. A. Shaikh, Using a one-class compound classifier to detect in-vehicle network attacks, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2018, Kyoto, Japan, July 15-19, 2018, pp. 1926-1929. doi: 10.1145/3205651.3208223.

[10] H. M. Song, J. Woo e H. K. Kim, In-vehicle network intrusion detection using deep convolutional neural network, Veh. Commun., vol. 21, 2020. doi: 10.1016/j.vehcom.2019.100198.

[11] E. Seo, H. M. Song e H. K. Kim, GIDS: GAN based Intrusion Detection System for In-Vehicle Network, in: Proceedings of the 16th Annual Conference on Privacy, Security and Trust, PST 2018, Belfast, Northern Ireland, Uk, August 28-30, 2018, pp. 1-6. doi: 10.1109/PST.2018.8514157.

[12] Robert Bosch GmbH, CAN specification - version 2.0, 1991.

12.Appendix A

Here we report some tables that comprehends more accurate measures on models' behaviour.

Legend:	
🟦	= CNN
🟧	= RNN
⬜	= multi CNN
🟩	= multi RNN

Table 2

Accuracy of models' prediction on OTIDS dataset

Window Size		8	16	32	64	128
No attack	CNN	29,55	44,32	91,39	73,47	79,66
	RNN	54,27	80,28	96,16	91,54	98,15
	CNN Fusion	57,28	58,15	88,8	84,85	90,18
	RNN Fusion	49,07	85,28	98,15	99,79	98,58
DoS attack	CNN	99,64	99,82	100	100	100
	RNN	99,88	99,95	100	100	100
	CNN Fusion	99,69	99,86	99,79	100	100
	RNN Fusion	99,86	99,95	100	100	100
Fuzzy attack	CNN	88,83	96,11	99,18	95,09	97,01
	RNN	95,56	98,56	99,47	99,86	100
	CNN Fusion	87,14	98,97	97,19	99,72	100
	RNN Fusion	94,79	98,1	99,72	99,79	99,86
Impersonation attack	CNN	70,29	82,01	74,15	93,31	99,72
	RNN	80,78	85,72	97,08	100	99,72
	CNN Fusion	68,43	91,08	91,64	97,8	98,44
	RNN Fusion	70,89	67,27	94,81	99,5	100

Table 3

Accuracy of GAN and Combined Method prediction on GIDS dataset

Window Size		8	16	32	64	128
No attack	CNN	60,06	80,3	99,89	99,22	95,31
	RNN	98,71	81,76	100	100	100
	CNN Fusion	87,85	81,1	100	100	94,74
	RNN Fusion	82,12	82,88	99,93	99,79	100
DoS attack	CNN	79,48	99,8	100	100	99
	RNN	47,78	99,98	100	100	100
	CNN Fusion	63,17	99,79	99,93	96,02	99,57
	RNN Fusion	64,36	95,86	100	100	100
Fuzzy attack	CNN	67,19	55,88	53,17	54,34	57,18
	RNN	53,6	62,04	54,87	55,12	54,91
	CNN Fusion	47,3	57,92	53,84	54,98	60,17
	RNN Fusion	62,37	61,55	52,42	55,69	56,33
Gear attack	CNN	82,14	99,93	100	100	100
	RNN	56,32	99,96	100	100	100
	CNN Fusion	65,43	99,8	100	96,51	99,86
	RNN Fusion	70,4	95,18	100	100	100
Rpm attack	CNN	78,81	99,91	100	100	100
	RNN	48,36	100	100	100	100
	CNN Fusion	60,66	99,82	100	96,87	99,43
	RNN Fusion	68,4	96,41	100	100	100

Table 4

Precision (P), Recall (R) and F1 Score (F1) of models' prediction on OTIDS dataset

Window Size	No attack			DoS attack			Fuzzy attack			Impersonation attack		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
8	0.5	0.3	0.37	1	1	1	0.79	0.89	0.84	0.54	0.7	0.61
16	0.69	0.44	0.54	1	1	1	0.87	0.96	0.91	0.65	0.82	0.73
32	0.77	0.91	0.84	1	1	1	0.96	0.99	0.98	0.94	0.74	0.83
64	0.87	0.73	0.79	1	1	1	0.98	0.95	0.96	0.79	0.93	0.86
128	0.96	0.80	0.87	1	1	1	0.97	0.97	0.97	0.85	1	0.92
8	0.68	0.07	0.12	0.84	1	0.91	0.38	0.99	0.55	0.73	0.1	0.17
16	0.87	0.58	0.7	1	1	1	0.95	0.99	0.97	0.7	0.91	0.79
32	0.89	0.89	0.89	1	1	1	1	0.97	0.99	0.89	0.92	0.9
64	0.97	0.85	0.91	1	1	1	1	1	1	0.87	0.98	0.92
128	0.98	0.90	0.94	1	1	1	0.98	1	0.99	0.93	0.98	0.96
8	0.73	0.54	0.62	1	1	1	0.94	0.96	0.95	0.65	0.81	0.72
16	0.84	0.8	0.82	1	1	1	0.99	0.99	0.99	0.81	0.86	0.83
32	0.97	0.96	0.96	1	1	1	1	0.99	1	0.96	0.97	0.97
64	1	0.92	0.96	1	1	1	1	1	1	0.92	1	0.96
128	1	0.98	0.99	1	1	1	1	1	1	0.98	1	0.99
8	0.63	0.49	0.55	1	1	1	0.9	0.95	0.92	0.61	0.7	0.66
16	0.72	0.85	0.78	1	1	1	0.96	0.98	0.97	0.85	0.67	0.75
32	0.95	0.98	0.96	1	1	1	1	1	1	0.98	0.95	0.96
64	0.99	1	1	1	1	1	1	1	1	1	1	1
128	0.99	0.3	0.46	1	1	1	1	1	1	0.59	1	0.74
CNN OHV	0.95	0.89	0.92	1	1	1	1	1	1	0.9	0.96	0.93
RNN OHV	0.92	0.93	0.93	1	1	1	0.99	1	0.99	0.94	0.92	0.93

Table 5

Precision (P), Recall (R) and F1 Score (F1) of models' prediction on GIDS dataset

Window Size	No attack			DoS attack			Fuzzy attack			Gear attack			RPM attack		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
8	0.54	0.58	0.56	0.95	0.33	0.49	0.62	0.59	0.6	0.35	0.75	0.48	0.84	0.42	0.56
16	0.43	0.98	0.6	0.99	0.46	0.63	0.95	0.54	0.69	0.66	0.81	0.73	0.96	0.45	0.62
32	0.51	0.89	0.65	0.96	0.51	0.67	0.94	0.57	0.71	0.68	0.98	0.8	0.82	0.55	0.66
64	0.51	0.96	0.66	0.56	0.82	0.66	0.92	0.59	0.72	0.66	0.35	0.45	0.98	0.46	0.63
128	0.47	0.97	0.63	0.97	0.52	0.68	0.96	0.56	0.71	0.67	0.9	0.76	0.96	0.46	0.63
8	0.48	0.66	0.56	0.98	0.41	0.58	0.81	0.55	0.65	0.37	0.75	0.49	0.9	0.42	0.57
16	0.45	0.94	0.61	0.98	0.45	0.62	0.9	0.56	0.69	0.6	0.82	0.69	0.96	0.46	0.62
32	0.5	1	0.67	0.97	0.54	0.69	1	0.55	0.71	0.69	0.99	0.81	1	0.45	0.62
64	0.49	1	0.66	1	0.49	0.66	1	0.55	0.71	0.68	1	0.81	0.99	0.46	0.62
128	0.5	0.98	0.66	0.97	0.55	0.71	1	0.56	0.72	0.69	1	0.82	1	0.46	0.63
8	0.34	0.98	0.5	1	0.49	0.66	0.94	0.51	0.67	0.66	0.41	0.51	0.94	0.43	0.59
16	0.44	1	0.61	1	0.53	0.7	0.99	0.55	0.71	0.68	0.82	0.74	1	0.45	0.62
32	0.5	1	0.67	1	0.55	0.71	1	0.55	0.71	0.69	1	0.82	1	0.46	0.63
64	0.5	1	0.67	1	0.55	0.71	1	0.55	0.71	0.69	1	0.82	1	0.45	0.62
128	0.5	1	0.67	1	0.55	0.71	1	0.56	0.72	0.69	1	0.82	1	0.45	0.62
8	0.47	0.65	0.55	0.97	0.5	0.66	0.81	0.53	0.64	0.39	0.78	0.52	0.93	0.43	0.59
16	0.45	0.97	0.62	0.99	0.54	0.7	1	0.55	0.71	0.64	0.82	0.72	0.97	0.46	0.62
32	0.5	1	0.67	1	0.55	0.71	1	0.55	0.71	0.69	1	0.82	1	0.46	0.63
64	0.5	1	0.67	1	0.55	0.71	1	0.56	0.72	0.69	1	0.82	1	0.46	0.63
128	0.51	1	0.67	0.89	0.59	0.71	1	0.56	0.72	0.7	0.93	0.8	1	0.47	0.63
CNN OHV	0.5	0.99	0.67	0.96	0.55	0.7	0.99	0.56	0.71	0.69	0.98	0.81	1	0.46	0.63
RNN OHV	0.5	0.98	0.66	1	0.55	0.71	1	0.56	0.71	0.68	1	0.81	1	0.46	0.63

Table 6
Accuracy of GAN and Combined Method prediction on GIDS dataset

	DCGAN	WGAN	WGAN-GP	CNN OHV + DCGAN	CNN OHV + WGAN	CNN OHV + WGAN-GP
No attack	97,01	98,08	97,65	100	99,86	99,86
DoS attack	54,55	57,82	62,23	100	100	100
Fuzzy attack	56,12	56,33	56,76	56,61	56,47	56,9
Gear attack	61,45	55,12	61,81	100	100	100
Rpm attack	54,2	50,78	56,47	100	100	100

Table 7
Precision (P), Recall (R) and F1 Score (F1) of GAN models' prediction

Dataset	Method	No attack			Attack		
		P	R	F1	P	R	F1
OTIDS	DCGAN	0.25	0.97	0.4	0.78	0.04	0.07
	WGAN	0.38	0.99	0.55	0.99	0.46	0.63
	WGAN-GP	0.5	0.98	0.66	0.99	0.68	0.8
	DCGAN combined	0.96	1	0.98	1	0.99	0.99
	WGAN combined	0.96	1	0.98	1	0.99	0.99
	WGAN-GP combined	0.96	1	0.98	1	0.99	0.99
GIDS	DCGAN	0.36	0.97	0.52	0.99	0.57	0.72
	WGAN	0.35	0.98	0.52	0.99	0.55	0.71
	WGAN-GP	0.38	0.98	0.54	0.99	0.59	0.74
	DCGAN combined	0.7	1	0.82	1	0.89	0.94
	WGAN combined	0.7	1	0.82	1	0.89	0.94