A survey of Intrusion Detection Systems for Controller Area Networks and FPGA evaluation

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Author: **Fabio Nappi**

Student ID: 928245
Advisor: Prof. Michele Carminati
Co-advisors: Stefano Longari
Academic Year: 2020-21

Abstract

In the past few decades car manufacturers produced and distributed vehicles equipped with a continuously growing number of electronic components, which allow modern vehicles to provide appealing features to the customers. These components, mainly sensors, actuators and Electronic Control Units (ECUs), require intra-vehicle networks to share their information: Controller Area Network (CAN) is the *de facto* standard in the automotive sector. This protocol was designed many years ago and security was not taken into account because cars were isolated systems at that time. However modern vehicles, despite using the CAN, are more and more connected to the outside world thanks to the V2X technologies, which raise the automotive security threat exposing new attack surfaces. Therefore securing the CAN is of critical importance, to this end literature propose many mechanisms to detect possible intrusions, i.e. Intrusion Detection Systems (IDSs). This thesis offers a detailed review of the latest studies of IDSs for CAN, their strengths and flaws; we propose this part as an extension and an update of Al-Jarrah et al. [4], also providing comparisons with it. Moreover we carry out an analysis of Field Programmable Gate Array (FPGA) systems to address the latency issue that affects most of the solutions found in literature: the use of complex algorithms makes it difficult to keep up real-time traffic. The analysis concludes providing an estimation of the hardware acceleration that could be offered by FPGA platforms to *CANova*, a modular IDS designed for CAN.

Keywords: Security, Controller Area Network, Intrusion Detection System, Survey, Field Programmable Gate Array, Latency

Abstract in lingua italiana

Negli ultimi decenni le case automobilistiche hanno prodotto e distribuito veicoli equipaggiati con un numero sempre maggiore di componenti elettronici, i quali consentono alle auto di fornire funzionalità d'interesse per i consumatori. Questi componenti, principalmente sensori, attuatori ed ECU, hanno bisogno di una rete interna al veicolo per condividere le informazioni: CAN è lo standard di fatto nel settore automobilistico. Questo protocollo è stato progettato molti anni fa e poiché all'epoca le automobili erano dei sistemi isolati, la sicurezza non è stata presa in considerazione. Nonostante l'utilizzo di CAN, i veicoli moderni sono sempre più connessi al mondo esterno grazie alle tecnologie V2X, le quali aumentano i rischi di sicurezza poiché offrono nuove superfici d'attacco. Perciò proteggere la rete CAN è di importanza cruciale, la letteratura propone diversi meccanismi per rilevare le possibili intrusioni, tra questi ci sono gli IDS. Questa tesi mostra una revisione dettagliata degli studi più recenti riguardo gli IDS per CAN, i loro punti di forza e le loro debolezze; questa parte viene proposta come un'estensione e un aggiornamento del lavoro di Al-Jarrah et al. [4], fornendo anche delle comparative con esso. Successivamente si conduce un'analisi dei sistemi FPGA mirata ai problemi di latenza che colpiscono la maggior parte delle proposte trovate nella letteratura: infatti con l'impiego di algoritmi complessi è più difficile mantenere il ritmo dei messaggi in tempo reale. L'analisi si conclude con una stima dell'accelerazione hardware che può essere fornita dalle FPGA a *CANova*, un IDS modulare progettato per CAN.

Parole chiave: Sicurezza, Controller Area Network, Intrusion Detection System, Indagine, Field Programmable Gate Array, Latenza

Contents

Abstract **i**

Abstract in lingua italiana **iii**

Contents **v**

1 Introduction **1**

2 Overview of Controller Area Network **5**
2.1 CAN Protocol 5
2.2 CAN Frames 6
2.2.1 Frame Types 6
2.2.2 Frame Format 7
2.3 CAN Security 8
2.3.1 Countermeasures against CAN attacks 9

3 Intrusion Detection Systems for CAN **11**
3.1 Categories of CAN IDSs 12
3.1.1 Flow-based IDSs 14
3.1.2 Payload-based IDSs 18
3.1.3 Hybrid IDSs 20
3.2 Features and Feature Selection 23
3.3 Datasets 25
3.4 Examined Attack Types 26
3.5 Evaluation Metrics 27
3.6 Benchmark Models 30

4 Research Gaps **31**
4.1 Detection Techniques and Placement 31
4.2 Features and Feature Selection 36

4.3 Attack Types and Reactive Measures 36
4.4 Performance Metrics and Benchmark Models 37

5 Field Programmable Gate Array for IDSs 39
5.1 FPGA Overview 39
5.2 FPGA for CAN IDSs 41
5.2.1 FPGA Implementation Examples 41
5.2.2 Examples Analysis 44
5.3 FPGA for CANova 45

6 Conclusions 49

Bibliography 51

A Appendix A 61

List of Acronyms 65

List of Figures 69

List of Tables 71

1 | Introduction

The technological progress of the last few decades guided the automotive companies to manufacture modern vehicles not only with mechanical components but also with electronic devices. These allow modern vehicles to provide new features that make them safer (*e.g.* airbags, emergency brake assist), more comfortable (*e.g.* cruise control) and even more entertaining thanks to infotainment systems.

Sensors, actuators, ECUs and communication devices are the electronic components integrated in modern vehicles. In particular the number of ECUs went through a substantial increase to enable the previously mentioned features on modern cars, the reason is that each ECU delivers a single specific functionality. Intra-vehicle networks allow the information exchange among different ECUs, in particular, as depicted in Figure 1.1, the ECUs are arranged into subnetworks based on their functionalities and these subnetworks can communicate with each other through the use of several gateways.

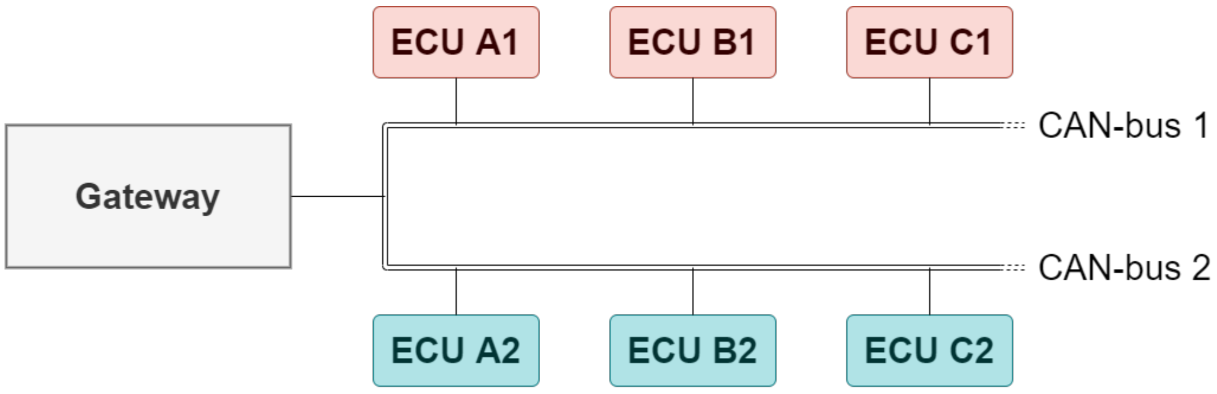

Figure 1.1: CAN structure

Because of its cheapness, reliability and real-time capabilities, CAN is the *de facto* standard network protocol that ECUs use to communicate with each other. However when CAN was designed (in 1983), the designer did not observe any security principle, indeed the idea that the network could be externally accessible was not taken into account. At that time vehicles were isolated systems, but nowadays vehicles are connected to the outside world via many communication interfaces (*e.g.* Bluetooth).

These communication interfaces lead to bigger attack surfaces, which, in addition to missing security features, translate into significant security threats. CAN vulnerability to cyber attacks has been widely demonstrated; for example Miller and Valasek describe their attacks in [51] and [50]. The missing message authentication and broadcast transmission are the main security issues of the CAN protocol that allow intruders to deliberately send messages on the CAN bus.

There are two types of security countermeasures that could be applied to manage CAN vulnerabilities: proactive and reactive ones. The first method makes use of systems, such as encryption, preventing cyber attacks; the latter instead is based on systems recognizing cyber attacks while they happen. Although these countermeasure methods are not mutually exclusive, the low computational power available on vehicles makes the proactive systems not a suitable solution. The most considered reactive countermeasure for vehicular networks is the IDS: a technique able to detect attacks when they occur in intra-vehicle networks.

The thesis proposes itself as an extension and a renovation of Al-Jarrah et al. [4]. Indeed this paper examines articles published after Al-Jarrah et al. review, but also focuses on the detection latency issue and studies the application of IDSs on hardware supports in order to exploit hardware acceleration advantages (in particular low latency and reduced power consumption).

The main contributions of the thesis are the following:

- Providing an overview of the CAN protocol, showing its properties and the security flaws.
- Providing a classification for latest studies of CAN IDSs based upon the type of used features and the employed detection technique.
- Showing the research advancement comparing our survey outcomes with the ones of Al-Jarrah et al. [4].
- Providing a comprehensive summary of the analyzed papers discussing intra-vehicle IDSs (Tables 4.1 to 4.3).
- Discussing current gaps in the intra-vehicle IDS research field, with a particular focus on the detection latency issue.
- Proposing a theoretical solution for the latency issue exploiting hardware supports.

The rest of the thesis is structured as follows: in Chapter 2 we present an overview of the necessary knowledge to understand this paper. Chapter 3 provides the review of the researches analyzed and categorizes them into three main classes: flow-based, payload-based and others/hybrid. Chapter 4 identifies current research gaps faced in the reviewed papers. Then Chapter 5 gives an overview of FPGA platforms, their benefits for CAN IDS research field and estimates the performance improvement for the *CANova* model. Finally we present our conclusions in Chapter 6.

2 | Overview of Controller Area Network

In this chapter we introduce the main concepts necessary to understand the thesis, that is the design and properties of CAN as well as an overview of CAN frames.

2.1. CAN Protocol

The CAN, also known as CAN-bus, is a serial, message-oriented communication protocol that allows the connection among different ECUs of the same vehicle. It was designed for the first time in 1983 by *ROBERT BOSCH GmbH* [72] and later standardized and revised in the standard ISO 11898-1 (2015). It is a very successful communication protocol, indeed it became the *de facto* standard for vehicle networks, but it is also used for other purposes.

The notable features of the CAN protocol are:

- **broadcast channel**: the CAN-bus uses the broadcast message routing scheme, i.e. any node can see and access any message sent in the network. In this way the transmitting node just needs to send the message on the bus, then the other nodes in the network decide to process it (or not) depending on the CAN Identifier (CAN-ID) of the message [20].

- **priority-based bus access**: each node can access the CAN-bus any time it is idle, but in case multiple nodes want to communicate at the same time, an arbitration mechanism based on priorities is employed: the message with higher priority is transmitted. Message priority depends on its CAN-ID, lower CAN-ID means higher priority; therefore “0” is the highest priority achievable [20].

- **robustness**: CAN can be considered a robust protocol, indeed it presents five error checking mechanisms (operating at different levels) which keep the bus running. Three of them work at message level and two at bit level. Failing any of these checks means the generation of an error message (see Section 2.2.1). Furthermore, in case of repeated errors (i.e. error limit threshold is reached), the failing node is disconnected from the CAN-bus [32].

- **message request**: each node can request the transmission of a precise message from another node by sending a Remote message (see Section 2.2.1), i.e. a message with a specific flag [20].

It is also worth mentioning that the original version of CAN supports a baud rate higher than $125\frac{Kbit}{s}$ (later versions can even reach $1\frac{Mbit}{s}$), and that bit encoding uses the "0" dominant and "1" recessive bits; where *dominant* means voltage put on bus while *recessive* means no voltage on bus. Therefore in case of collisions the dominant bit gets through.

Another important feature of CAN is the so called *bit stuffing* mechanism. This mechanism inserts a dominant bit after 5 consecutive recessive bits and inserts a recessive bit after 5 consecutive dominant bits. If 6 consecutive bits are dominant or recessive, the frame containing this sequence is discarded by other nodes and then these transmit an error frame (see Section 2.2.1).

Summarizing, the CAN is simple, cheap, centralized, robust and efficient. These properties make it a qualified candidate to operate in a real-time environment such as intra-vehicle networks.

2.2. CAN Frames

As we said before, CAN is a *message*-oriented communication protocol; so in this section we will go into detail of its messages. We define CAN messages as *frames*.

2.2.1. Frame Types

There exist four different types of frames that can be transmitted on the CAN-bus:

- **Data frame**: the most common type of frame on the bus, it carries information for other ECUs.
- **Remote frame**: it is used to request data from another ECU, it carries no data.
- **Error frame**: it is sent by ECUs that detect an error in another frame. It causes the transmitter of the malformed frame to send it again.
- **Overload frame**: it is used to signal that an ECU is becoming too busy. It is sent by the overloaded ECU itself to add some extra delay among the following messages.

Frames can be transmitted on the CAN-bus only after an Inter-Frame Space (IFS), which is used by the protocol to avoid collisions. In this thesis we will analyze only Data frames as they constitute the almost totality of frames transmitted on CAN.

2.2.2. Frame Format

There exist two frame formats: the original one and the extended one. When CAN was designed the authors thought that 11 bits for identifying the frame were enough, but the increasing number of ECUs and their features led to an extension of the original structure with additional bits to identify other functionalities. Next we will present the structure of original frame format, the reason is that the extended format has just additional bits for the CAN-ID field.

The original structure of frames consists of different fields, as represented in Figure 2.1.

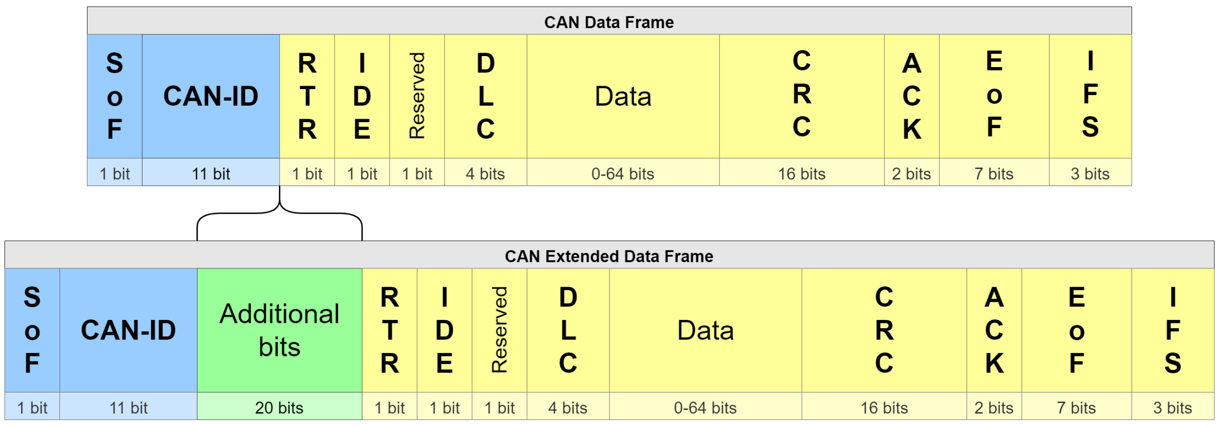

Figure 2.1: Original (top) and extended (bottom) CAN frame format, extracted from [72].

The following list describes each field of the standard structure.

- **Start of Frame (SoF)**: a single dominant bit that signals the start of the frame.
- **CAN-ID**: 11 bits representing the feature transmitted in the frame. The message priority depends on this field, the lower the value the higher the priority.
- **Remote Transmission Request (RTR)**: a single bit to distinguish between Remote and Data frames. Dominant bit stands for Data frame, recessive bit means Remote frame.
- **Identifier Extension Bit (IDE)**: a single bit to distinguish between standard and extended CAN frame formats. Dominant bit means standard frame format.
- **Reserved**: a single reserved bit, should be dominant.

- **Data Length Code (DLC)**: 4 bits specifying the number of bytes in the Data field, thus Data length goes from 0 to 8 bytes.
- **Data**: from 0 to 64 bits with 8-bits steps. It contains the information that the message wants to transmit. Multiple values can be packed inside a single Data field, we define them *signals*. Signals meaning is kept confidential by vendors because it is part of the *Security by Obscurity* approach that will be analyzed in Section 2.3. The length inconstancy is handled by the CAN protocol employing the DLC field.
- **Cyclic Redundancy Check (CRC)**: 16 bits representing the checksum (number of bits transmitted) from SoF to Data field. It is used by other ECUs to detect transmission errors. Last bit is a delimiter, should be recessive.
- **ACKnowledge (ACK)**: 2 bits. The first bit is set recessive by the sender node, and then overwritten with a dominant bit by the receiver. The other bit is a delimiter, should be recessive.
- **End of Frame (EoF)**: 7 recessive bits that signal the end of the frame.
- **IFS**: 3 recessive inter-frame bits.

2.3. CAN Security

The CAN was designed for isolated systems, therefore security was not taken into account during its development. The features it provides, reported in section 2.1, are suitable for vehicle's real-time requirements, but at the same time they are source of security vulnerabilities. Attackers can easily access the CAN-bus either physically or remotely: On-Board Diagnostics (OBD) is a way for a physical attack as long as Wi-Fi or Bluetooth are means for remote attacks. Indeed modern vehicles are equipped with both those wireless interfaces and also with OBD, which is a computer-based system that can access the CAN via a physical connection: it is commonly used to show emissions and perform some troubleshooting [15].

We detailed below the most important security lacks of the CAN-bus with respect to security's basic principles: *confidentiality*, *integrity*, *availability* and *non-repudiation*.

- **missing encryption**: CAN cannot adopt message encryption of any type because it has to keep its simplicity and speed to stick with real-time computing requirements. This absence violates the confidentiality principle, allowing attackers to sniff messages from the network.
- **missing authentication**: the CAN protocol also lacks of any authentication sys-

tem, violating this way the non-repudiation principle; attackers can easily inject messages in the network impersonating another ECU.

- **missing integrity checks**: CAN-bus misses data integrity checks. The CRC field of a CAN frame can be tampered without much effort because the standard explains how to compute it. So attackers can manipulate any frame setting the right CRC value. This vulnerability violates the integrity and availability security principles.

The combination of the vulnerabilities listed above allows hackers to misuse the protocol performing different types of attack, *e.g.* a Denial of Service (DoS) attack can be easily achieved injecting messages with high priority at high frequency.

The vendors voluntarily hide the Communication Database for CAN (DBC) file trying in this way to make attacks more complicated for hackers. Anyway the *Security by Obscurity* method has been rejected by security experts many years ago in place of *Security by Design*, in fact the first approach alone is not enough for avoiding attacks.

2.3.1. Countermeasures against CAN attacks

In front of the CAN security threats described above, it is necessary to apply a defending strategy to protect intra-vehicle networks from external attacks. There exist two main countermeasure classes: the proactive and the reactive one. Their lifecycles are exposed in Figure 2.2.

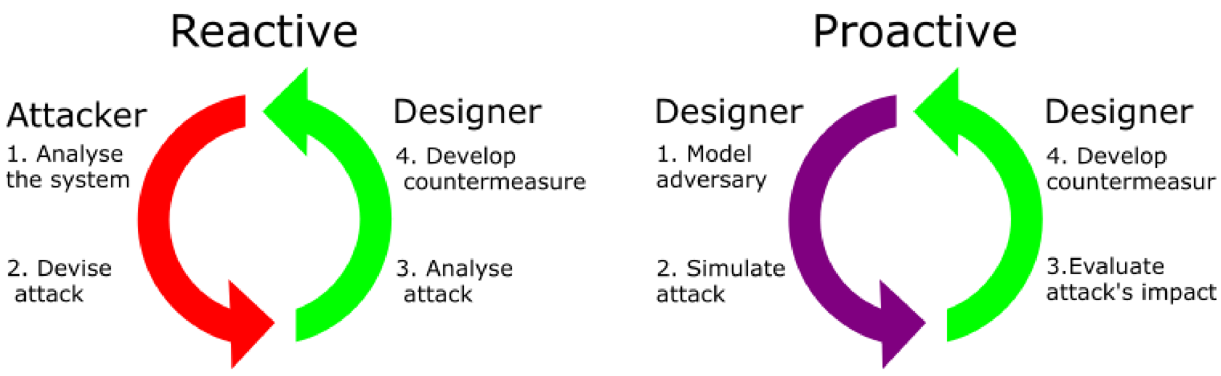

Figure 2.2: Reactive and proactive countermeasure lifecycles, extracted from [15].

The aim of proactive countermeasures is to avoid that an attack occurs, while the goal of reactive countermeasures is to perform some actions when an attack is detected. The designer of a proactive countermeasure firstly models a hypothetical adversary and simulates an attack, then finds the best solution to avoid the attack itself. Instead the designer of a reactive countermeasure firstly detects an attack and then finds a method to make it harmless.

In literature, the most used countermeasure type for CAN-bus security is the reactive one, because it is easier to implement and usually requires less computational power with respect to the proactive alternative. In addition proactive countermeasures change the system design, at the opposite of reactive ones which can be added without modifying the network. Among the different existing reactive countermeasures, the IDS is the most common solution; this is the reason why part of this thesis focuses on a survey of IDSs for securing CAN.

3 | Intrusion Detection Systems for CAN

IDSs are usually classified with respect to the employed detection technique into *knowledge*-based or *anomaly*-based IDSs.

The formers use some kind of knowledge to understand if an intrusion is happening, *e.g.* a database with patterns of already known attacks, also called *signatures*. Thus a pattern matching process, for each inspected sample, over the whole database is required to detect intrusions. These operations do not scale well: having a large database implies high resource requirements in terms of Central Processing Unit (CPU), memory and also power consumption. Another significant issue for knowledge-based IDSs is the limited detection ability of novel attacks (especially zero-day attacks), because they are not included in the knowledge possessed by the system. Database updates should be performed frequently to mitigate this problem. On the other hand, knowledge-based IDSs are characterized by a low False Positive Rate (FPR) as they are triggered only by already known attacks [4].

At the opposite there are anomaly-based IDSs: they compare the inspected traffic with the normal behavior of the system to identify intrusions, which are remarkable irregularities. In order to identify the normal behavior of the system, it is necessary to have a sufficient amount of attack-free data, which is not always available. Moreover they are characterized by a relatively high FPR, as benign outliers can trigger them. On the other hand, anomaly-based IDSs are able to detect novel attacks and do not need a storage for attack signatures, which means that they require less memory than the knowledge-based alternative [4].

Most of the IDSs reviewed in our survey belongs to the anomaly-based category, the main reason is that securing CAN is a problem requiring the identification of many and (too) complex patterns necessary to the pattern matching process, incurring in the resource limits issue previously mentioned. In addition, we are dealing with an evolving problem: during the lifetime of a vehicle (about 10 years), novel attacks are likely to be disclosed.

3.1. Categories of CAN IDSs

In the current section we will review the analyzed IDSs following the classification principles used by Al-Jarrah et al. [4], thus we will categorize IDSs with respect to two of their characteristics, as follows:

1. **features type**: the main classification is performed depending on the type of features employed by the authors of the detection device. We identified three main classes:
   - *Flow-based*: belong to this category those IDSs that use features not related to the contents of CAN messages.
   - *Payload-based*: belong to this category those IDSs that use features derived from the contents of CAN messages.
   - *Hybrid*: belong to this category those IDSs that use both features derived from the contents of CAN messages and those not related to it.
2. **detection technique**: the subclassification is performed depending on the category to which the implemented detection technique belongs. We recognized these categories:
   - *Rule-based*: the detection technique compares CAN traffic with predefined rules. If messages match (or break, depending on the design) a rule, the IDS reports an intrusion.
   - *Time and Frequency Analysis (T&FA) -based*: the detection technique performs time and/or frequency based analysis on CAN traffic to recognize intrusions.
   - *Computational Intelligence and Information Theory (CI&IT) -based*: the detection technique belongs to the Computational Intelligence or Information Theory research fields (*e.g.* Machine Learning (ML)).
   - *Hybrid*: the detection technique belongs to more than one of the previous categories.
   - *Others*: the detection technique does not belong to any of the previous categories.

The classification model is also represented in Figure 3.1.

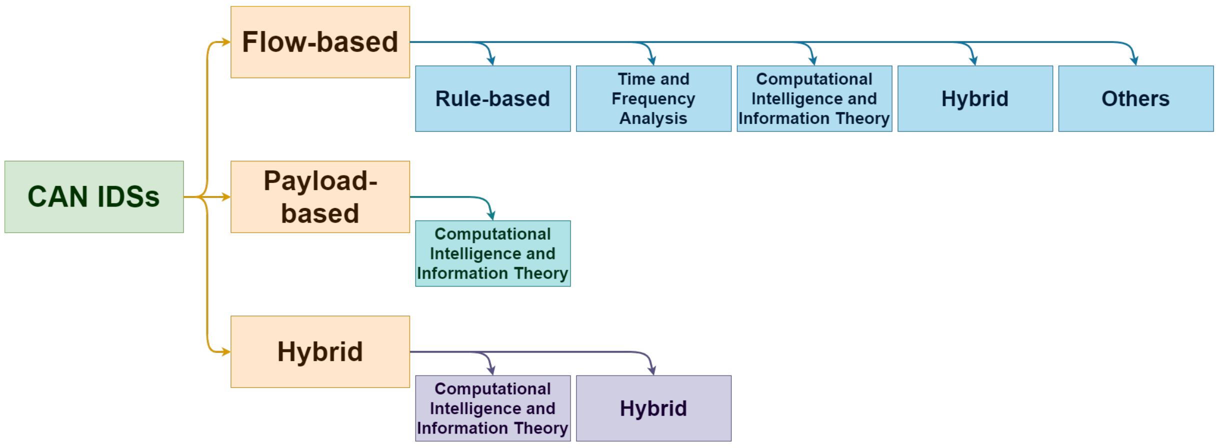
CAN IDSs
Flow-based
Rule-based
Time and Frequency Analysis
Computational Intelligence and Information Theory
Hybrid
Others
Payload-based
Computational Intelligence and Information Theory
Hybrid
Computational Intelligence and Information Theory
Hybrid

Figure 3.1: Categories of CAN IDSs.

Being an extension of Al-Jarrah et al. [4], this survey reports papers published after its publication date: in particular this thesis covers papers published between the beginning of 2019 and the end of 2021. Unlike the original survey, we will emphasize the detection latencies obtained in each reviewed paper, indeed we believe that latency is one of the most important metrics to take into account when designing a device for an environment with such strict timing requirements like intra-vehicle networks.

Another metric that we will use in the reviews is the F1-score, also known as F-score. The formula of F1-score comes from precision and recall:

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{3.1}$$

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.2}$$

$$F_1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{3.3}$$

We prefer to use this metric to briefly describe the model's performances rather than accuracy because accuracy does not take into account class imbalances in the dataset. Indeed most of the datasets used in literature present much less attack messages than normal messages, as it should also be in a real-world scenario. However, as mentioned in Section 3.5, the evaluation of a model should not entirely rely on this only metric. The Matthews Correlation Coefficient (MCC) would probably be a better way to compare the proposed models, but only 1 paper among the reviewed ones makes use of it; while F1-score can be found in almost every document.

Despite our effort, this survey may not cover the entire research of CAN IDSs.

3.1.1. Flow-based IDSs

In our survey we found many Flow-based IDSs. Most of them employ detection techniques from T&FA and CI&IT categories, but we found also a Rule-based sample, a hybrid sample and a couple of models not belonging to any of the former categories.

Rule-based IDSs

Blevins et al. [12] implemented a Binning method that considers the number of messages transmitted within a certain time window. A message is considered malicious when the last 6 frames arrive in less than $\alpha_i\mu$ seconds, where $\alpha_i = (2+i)/2$ for $i = 1, ..., 18$ is the detection threshold and μ denotes the average inter-message time for each CAN-ID. Experimental evaluation shows that the proposed model performs at its best, referring to precision and recall metrics, with $\alpha = 3.5$. The detection latency of this technique ranges between 2 and 5 μ depending on the threshold (the minimum latency is obtained for $\alpha \geq 5$). This means that many other messages get transmitted meanwhile, which suggests that this IDS cannot be employed in real-world scenarios.

T&FA-based IDSs

Young et al. [86] developed an IDS that analyzes CAN message frequencies. Frequency was chosen in place of time-interval because the authors observed a more consistent behavior for frequency, in particular during the switch of vehicle's mode of operation. An intrusion alert is raised if the current frequency exceeds by 2 times the normal frequency, this threshold was obtained from their *message injection attack* simulation which required a frequency at least 2 times higher than the normal to succeed. It is worth noticing that the simulation was carried out via an OBD port. Experimental evaluation proved the proposed model as a qualified IDS. However there exist many types of CAN attacks, but the authors evaluated the model only against injection. There is no information about detection latency.

Olufowobi et al. [62] implemented a system building a timing model of CAN messages for each vehicle, indeed, as they state, even vehicles from same brand and model have different timing characteristics. Frames can be labeled as *delayed* and *dropped* respectively if they do not arrive by the maximum computed timestamp or if they do not arrive within the minimum timestamp of the next frame. Nevertheless this method can be only applied to those messages that show periodic behavior. The detection latency of this method ranges between 0 and 10 ms, but the authors do not describe the platform used for evaluation.

The IDS developed by Han et al. [27] is based on periodic event-triggered intervals of CAN messages. Ground truth of event-triggered intervals is measured for each CAN-ID: a message is considered abnormal if its event-triggered interval does not fall within the ground truth thresholds. The proposed model is also able to identify the type of attack when detected, this is accomplished computing different statistics and giving them in input to a ML model. The evaluation was performed on a platform powered by Intel Core i7-7500U with 16GB of RAM. The detection latency differs with respect to the size of time window used, it can range between 1 ms to 15 ms. We remind that using windows (except for backward sliding windows) is not the optimal solution as the IDS needs to wait for next frames before evaluating the first one.

Bozdal et al. [14] proposed a model based on wavelet coefficients of CAN frames. The frequency profile is obtained with the Continuous Wavelet Transform (CWT), a tool for the precise localization of frequency components on the time axis. Thresholds are computed by the anomaly decision module and compared with CAN traffic, values higher than the threshold are denoted as anomalies. The evaluation, performed with windows of 128 frames, highlights the strong potential of the IDS and registered a detection latency between 3-6 ms depending on attack types. However the authors do not specify the used platform.

CI&IT-based IDSs

Ning et al. [60] built an IDS based on voltage waveform of frames transmitted on CAN-bus. They divide each waveform into three parts: rising edge, falling edge and stationary segment, where the rising and falling edges contain most of the feature information. The used detection method is Local Outlier Factor (LOF), which is a distance-based anomaly detection technique: points (i.e. frames) with significant lower local density are considered outliers (i.e. malicious). The data was gathered from real vehicles as it is not reliable to simulate voltage waveforms: the authors used the OBD interface to access the CAN, an oscilloscope to save voltage data, *CANalyst II* to send physical attacks and *CANWIFI II* to send remote wireless attacks. No information about detection latency is provided.

The detection method developed by Song et al. [69] is based on Convolutional Neural Networks (CNNs). The proposed ML technique is fed with a squared matrix of $29 * 29 * 1$ dimensionality that contains the CAN-IDs of the last 29 frames. Indeed the model is applied on a CAN-bus that uses frame's extended format, where CAN-IDs are made up of 11 standard bits plus 18 additional bits (Section 2.2.2). The modeled CNN was called *Reduced Inception-ResNet*, indeed it has been designed simplifying the original *Inception-ResNet* which is fed with 299∗299∗3 image data. The evaluation shows a detection latency of 5 ms per sample (29 frames) on a platform powered by $2 * 2.30GHz$ Intel Xeon with Nvidia Tesla K80. The model can test about 5800 frames per second ($= 29frames * \frac{1}{0.005s}$), so it should accomplish real-world timing requirements.

Yu et al. [88] implemented a multiple Rényi entropy-based IDS. The authors provide analysis of entropies (Rényi included) with respect to changing attack rates, indeed only DoS and Fuzzy attacks are analyzed. Moreover they provide an algorithm for the estimation of Rényi entropies with orders 2, 3 and 4; significant changes in entropies are reported as attacks. No information about detection latency is provided.

Song and Kim [67] proposed a self-supervised anomaly detection model using noised pseudo normal data. The model is divided in two modules: a generator module based on Long Short-Term Memory (LSTM) that mimics real CAN traffic and is used as abnormal data for training and detection; an anomaly detection module, they used the CNN-based model previously reviewed [69]. The evaluation shows strong performances with respect to the other compared benchmark models. Although authors do not provide any information about detection latency, we can confidently suppose it is the same stated in [69].

Nam et al. [55] developed an IDS using a bi-directional Generative Pretrained Transformer (GPT) network. The detection method works as follows: to detect whether a given CAN-ID sequence shows evidence of attack behavior, it is provided as input to the bi-directional GPT network to evaluate its Negative Log-Likelihood (NLL). If the value of NLL is greater than a prespecified threshold, then an attack is identified. The model proves to be very effective in the evaluation process with an F-score above 95.2% in the Worst Case Scenario (WCS) (against spoofing attack). Unfortunately authors did not include detection latency among the multiple evaluation metrics.

Mehta and Meyer [49] proposed three tree-based learning algorithms, they refer to them as Decision Tree Ensemble-based Intrusion Detection System (DT-DS). The used algorithms were *Adaboost*, *gradient boosting* and *random forest*; each was tested with and without message profile in input. Indeed, as the dataset used is simulated, the authors have the message profile feature that reports which communication profile is sending the data. CAN attacks were performed with *Gaslighter*, a predictive attacker that predicts the arrival time and sequence number of the next message and injects a forged message, misleading the system to reject the genuine message and accepting the malicious one. It is worth noting that this IDS is developed and tested on aircraft's CAN traffic which is called *CANaerospace*: it is a simplification of standard CAN. No information about detection latency is provided by the authors.

Hybrid IDSs

Sunny et al. [71] designed a hybrid model based on the recurring patterns and time intervals of individual CAN-IDs. The key features of this design are: compatibility across different vehicles, faster response time and localized anomaly detection. The model is implemented through the use of a sliding window with different sizes (from 1 to 4 frames); during the so called *training*, unique CAN-ID sequences are stored in a tree data structure and time interval are stored too. During detection the CAN-ID sequences are compared with stored ones, if the pattern does not match any then it is flagged as anomaly; if the sequence is known then a check of the time interval is performed, if the check fails then it is flagged as anomaly. We classify this IDS as hybrid because its detection technique is a mixture of rules and T&FA. However the evaluation process proves the model not much effective: FPR increases too much in window size of 4, with small window sizes it produces much less false positives but the True Positive Rate (TPR) decreases too. The configuration with window size of 2, which is the only one suitable, has the best detection latency so far, with an average of about $0.05\frac{ms}{window}$ and $0.21\frac{ms}{window}$ in WCS. Furthermore, to emphasize these results, the evaluation was performed on a low-end device as a Raspberry Pi 3. Finally the use of a sliding window makes it able to operate in a real-world scenario.

Others IDSs

Islam et al. [33] proposed a statistical analysis -based IDS. The method consists of several steps: first of all CAN frames are transformed into a graph structure; then graph-based features are extracted to import for anomaly characterization; after that a hypothesis based on the safe population window is built; finally the test population window is compared to the base population. Specifically, values used in this last comparisons are chi-square values. Population window is defined as a set of windows and each window has a size of 200 messages. The model evaluation, carried out on a platform powered by $8*3.8GHz$ Intel Xeon with $32GB$ of RAM, shows a detection latency of about $258.9\frac{\mu s}{window}$; even if this latency is extremely low, the IDS has to wait for many frames before building the window and start the detection.

Aliyu et al. [5] proposed a Blockchain-based Federated Forest IDS (BFF-IDS) to address the problem of sensitive CAN data sharing. The method makes use of Federated Learning (FL) to create a random forest model starting from partially trained models provided by individuals, keeping in this way underlying data confidential. This IDS employs techniques coming from both T&FA (Fourier transformation is applied to CAN-ID cycles to translate them into the frequency domain) and CI&IT (the detection technique is a random forest), but we choose to classify it in *Others* rather than *Hybrid* because of its peculiar blockchain mechanism and distributed machine learning method. The evaluation was performed using different number of miners (i.e. partially trained models) ranging from 5 to 20: the best performance was obtained with the minimum number of miners, an F-score of about 0.97; the authors explain this result with the less training data available to more miners. Indeed the dataset used was always the same, so more subsets means smaller subsets. The paper provides detailed information about the platform used for the evaluation process, but no data related to detection latencies.

3.1.2. Payload-based IDSs

Here we give a brief review of Payload-based IDSs we have found. Unlike Al-Jarrah et al. [4] we found only Payload-based IDSs employing detection techniques belonging to the CI&IT category.

CI&IT-based IDSs

Zhang et al. [89] designed an anomaly-based IDS that uses Gradient Descent with Momentum (GDM) and Gradient Descent with Momentum and Adaptive Gain (GDM/AG) to improve efficiency and accuracy. The neural network is fed with vehicle CAN frames, in particular the features used are *speed*, *rpm*, *throttle pedal position* and others. Indeed this paper starts from the (weak) assumption of knowing what each CAN-ID stands for; this assumption is almost never satisfied in a real-world scenario because of the confidentiality of the DBC file. The detection latency of this IDS ranges between 2 and 4 ms on a platform powered by a 1.80GHz Intel Core i5 with 8GB of RAM.

The IDS built by Hossain et al. [30] is based on LSTMs. The proposed model is able to label CAN frames as *benign*, *DoS*, *Fuzzing* and *Spoofing* starting from their CAN-ID, DLC and the 8 bytes of the data field. The ML algorithm uses batch sizes of at least 256 frames, this means that the IDS has to wait for too many frames before executing the detection. The evaluation process highlights the effectiveness of the model and the authors provide much information about the platform used to build their model. Unfortunately they did not provide detection latency metrics as well.

Minawi et al. [52] propose a modular IDS based on multiple ML algorithms. The model consists of 3 layers: *CAN Message Input Layer*, *Threat Detection Layer* and *Alert Layer*. The *Threat Detection Layer* is constituted by multiple *Attack Specific Modules*, the authors propose four modules to detect gear spoofing, rpm spoofing, DoS and fuzzy attacks.

The chosen ML algorithms are Random Tree (RT), Random Forest (RF), Stochastic Gradient Descent (SGD) and Naive Bayes (NB), and the chosen inputs of these algorithms are CAN-ID and data field of CAN frames converted from hexadecimal to decimal values. The IDS was evaluated on a machine with $6 * 2.6GHz$ Intel Core i7 and $16GB$ of RAM. It resulted in a detection latency between $2\frac{\mu s}{frame}$ and $2\frac{ms}{frame}$.

Yang et al. [85] designed a multitiered IDS to detect known and unknown attacks on intra-vehicle networks. This ability is given by the presence of a signature-based IDS and an anomaly-based IDS. The paper describe accurately their implementation steps motivating their decisions. The features used were CAN-ID and just 3 out of 8 bytes from the data field of CAN frames: $data[5]$, $data[3]$ and $data[1]$. The model evaluation proves its strength: it reaches a F-score close to 1 for known attacks, while the average over unknown attacks is above 0.80. The authors provide also average detection latency information, which is approximately $0.5\frac{ms}{frame}$ on a low end device as a Raspberry Pi 3.

The IDS proposed by Pascale et al. [64] is a model based on Bayesian Networks able to understand if messages traveling on the CAN-bus are malicious or not. The model is trained from simulated data, which is (in general) not a good practice, especially nowadays that some real CAN datasets are published online. As the used dataset is simulated, the authors assume to know the meaning of each CAN-ID like in [89]; therefore *steering*, *speedometer*, *gear*, *brake* and many other basic data are known and used as inputs of the model. Furthermore they feed their model with other derived data (like acceleration) computed from the basic ones. It is worth noticing that the IDS is deployed on an embedded device like the *PYNQ-Z1* board, but no information about detection latency is given.

Mehedi et al. [48] developed an IDS based on a deep transfer learning model named *LeCun Network*. It is able to transfer the knowledge contained in the source domain data to the target domain and combine the target domain data to build an efficient IDS to improve the detection accuracy for any vehicle. The model is fed with the following features: CAN-ID, DLC and the data field of CAN messages. We have no information about detection latencies of this method.

Kang and Shen [35] developed an IDS based on a LSTM network able to detect both known and unknown abnormal messages. The model is fed with CAN-ID and the data field of CAN messages; fabricated frames are produced by a Generative Adversarial Network (GAN) to train the LSTM network. Unlike [48] this model has a network transfer learning module which is able to transfer a pretrained neural network from a vehicle to others. The authors describe the platform on which the evaluation was executed, but there is no data about detection latencies.

Freitas De Araujo-Filho et al. [21] propose an Intrusion Prevention System (IPS) using the Isolation Forest (IF) algorithm. An IPS is a tool that monitors a network (like an IDS) and prevents malicious activity performing some actions. The authors base their design on the peculiar assumption that malicious frames must be detected and discarded before their transmission is completed. The chosen mechanism to discard a malicious frame was *bit stuffing* (see Section 2.1), so malicious frames must be detected before their last 6 bits are transmitted. Therefore waiting for more bits to perform the detection implies less time for detecting. This means that on a standard CAN which has a speed of $500\frac{Kbit}{s}$, a bit is transmitted every $2\mu s$; for example, according to the authors, an IPS waiting until the fifth byte of the data field has only $86\mu s$ to detect a cyber-attack. The evaluation process proves that this model is one of the best solutions found, with an F-score above 0.97 and a detection latency shorter than $79\frac{\mu s}{frame}$ for the IF using just the first 5 bytes of the data field. Thus the model seems to be suitable for real-world application.

The IDS designed by Nazakat and Khurshid [58] can use either a MultiLayer Perceptron (MLP) or a Decision Tree (DT) as detection algorithm. These algorithms take in input the CAN-ID and the data field of CAN frames converted into decimal representation. The model evaluation carried out by the authors reached an accuracy of 100% for both the implementations; however such an impressive result may be biased by the test dataset which was completely written by the authors with MATLAB's *Vehicle Network Toolbox* and a DBC file. Finally the paper does not provide any information about detection latencies.

3.1.3. Hybrid IDSs

Now we talk about the Hybrid IDSs we reviewed in our survey. Most of the models found for this category of IDSs use CI&IT-based detection techniques, but we also found an instance making use of multiple techniques.

CI&IT-based IDSs

Khan et al. [38] developed an IDS based on a neural network composed by LSTM cells followed by a fully connected layer. The used datasets are coupled with a respective DBC file, so the proposed model knows the contained features. The authors used the Pearson Correlation Coefficient (PCC) to select a subset of features to feed the neural network. It is not clear the resulting subset, but some of these features are time-dependent. There is no data about detection latencies, however the results of the evaluation process are quite poor: the accuracy metric is always much lower than 90%.

Barletta et al. [8] proposed a model based on an unsupervised Kohonen Self-Organizing Map (SOM) network integrated with the K-means algorithm. The SOM technique is a nonlinear mapping network that aims at computing similarities among input data vectors. The used input vector contains the following values obtained from CAN frames: CAN-ID, DLC, data field and time interval. The proposed IDS reached an F-score of 1 for some attack types (DoS and Spoofing), but the metric goes down to about 0.93 on a dataset with mixed attacks. The authors do not specify any information about detection latencies.

The anomaly-based IDS presented by Park and Choi [63] employs a Multi-Labeled Hierarchical Classification (MLHC) technique. The design of this type of technique is depicted in Figure 3.2 alongside two similar approaches: the Single Layer Multi-class Classification (SLMC) and the Two Layers Multi-class Detection (TLMD). This structure makes it able to even classify the detected attack just like [27]. The MLHC takes in input the time interval, the CAN-ID, the DLC and the data field of CAN messages; in particular the data field is converted into a string of bytes as lengthy as stated by DLC value. The evaluation of the model shows its strengths: using the RF algorithm the IDS reaches a F-score higher than 0.999 and the average detection time is about $23\frac{\mu s}{frame}$. Furthermore the evaluation compares the MLHC technique against SLMC and TLMD: in both the comparisons the model using MLHC has better metrics, especially detection latency. Unfortunately the authors do not describe the platform used for evaluating.

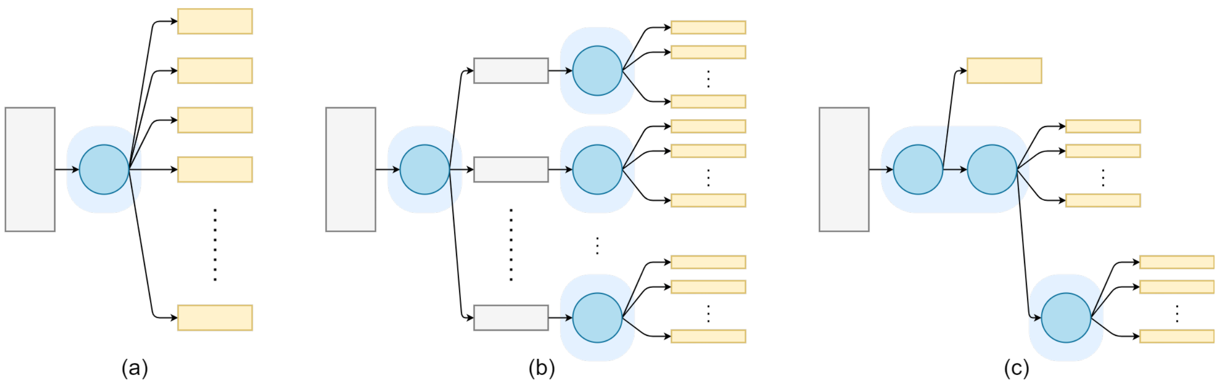

Figure 3.2: Classification methods comparison: (a) SLMC model; (b) TLMD model; (c) MLHC model [63].

Javed et al. [34] developed an approach called *CANintelliIDS*, a combination of CNN and Attention-based Gated Recurrent neural network (AGRU). The features sent in input to *CANintelliIDS* are: time interval, CAN-ID, DLC and the eight byte of the data field of CAN messages. However the performances of the model are not impressive: the F-score metric is always below 0.94 in both *single-attack* scenario and *mixed-attack* scenario. Finally the authors provide the specifications of the platform that executed the evaluation process, but no data about detection latencies is given.

He et al. [28] proposed an IDS based on a DT algorithm. In this model the detection algorithm is fed with the following features: time interval and Manhattan distance of the data field, both computed for respective CAN-IDs. The evaluation was performed on different datasets, each containing an attack type. The F-score in injection dataset is 1 and slightly lower (0.99) in camouflage dataset, but the value decreases to 0.90 in suspension dataset and 0.95 in tampering dataset. No information about detection latencies is provided.

The IDS developed by Narasimhan et al. [56] is based on a Gaussian Mixture Model (GMM). In order to learn the optimal features from CAN frames the authors employed the Improved Deep Embedded Clustering (IDEC) approach, which clusters them using an autoencoder and K-means respectively. The GMM helps to cluster frames data into normal and attack. The evaluation results show that this model is not very effective. The F-score was computed on different datasets: its maximum (0.80) is reached in the same dataset partially used for training; while in other datasets the value is always below 0.67. Such performances are not suitable for real-world scenarios. Finally detection latencies are not specified.

Balaji and Ghaderi [7] designed *NeuroCAN*, an IDS based on linear embeddings and LSTM units. In order to handle the different characteristics of CAN messages, the authors trained a separate network for each CAN-ID; in particular only the linear embedding layer is separated into different networks, indeed the LSTM layer is single and shared as represented in Figure 3.3. The model is fed with the data field of a CAN frame alongside the data field of others within the subsequent time step. The evaluation was performed on two datasets containing different spoofing attacks: the registered F-score goes from 0.95 to 1 between the two datasets. The authors do not provide any information about detection latencies.

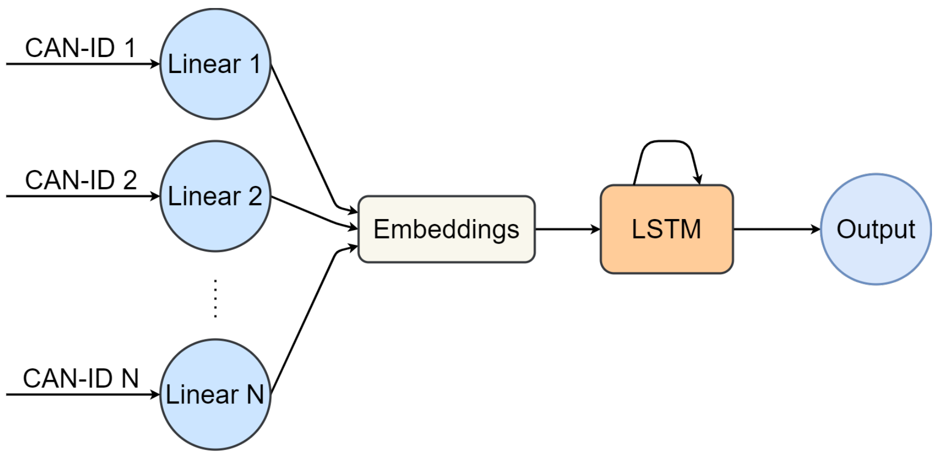

Figure 3.3: NeuroCAN network structure [7].

Hybrid IDSs

Nichelini and Pozzoli [59] proposed *CANova*, a classification-based modular IDS for CAN. The authors affirm that messages with different CAN-IDs have distinct characteristics, thus a single IDS for all of them will never have satisfying performances. Their modular structure allows to route each message through a sequence of IDS modules according to the CAN frame characteristics (inherited from the CAN-ID). The modules proposed by the authors are the following but others can be easily added: a flow-based, a rule-based, an Hamming distance -based, a Vector AutoRegression (VAR)-based and a Recurrent Neural Network (RNN) autoencoder-based module. The model is fed with CAN-IDs, time intervals and the data field of the CAN frame. The evaluation process was carried out using a real dataset modified with different types of attack, *e.g.* DoS, fuzzy, replay and others: the F-score reported by the authors varies depending on the attack type and goes from 0.6036 to 0.9684. About latency, the average Testing Time per Packet (TTP) ranges between 0.2113 to 0.2688ms on a high-end machine powered by a 3.70GHz Intel Core i7-8700K, 32GB of RAM and a *NVIDIA GeForce GTX 1080* Graphics Processing Unit (GPU).

3.2. Features and Feature Selection

One of the most important characteristics to build successful IDSs is the selection of proper features that will be used by the detection algorithm. We classify these features into two main categories:

- **physical features**: belong to this type those features describing the physical state of the system (*e.g.* speed, steering angle) [4].

- **cyber features**: belong to this type those features describing the communication and data aspects of the system (*e.g.* message sequences, time intervals) [4].

Analyzing the reviewed papers we found out that most of them (28 out of 31) feed their IDS with cyber features. Only two of them used physical features ([89] and [64]) an just one makes use of both types of features ([38]). A summary of these results is represented in Figure 3.4.

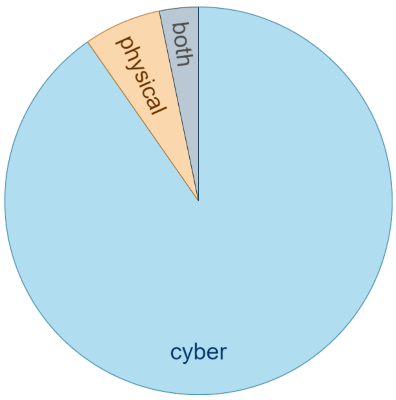

Figure 3.4: Distribution of feature sets.

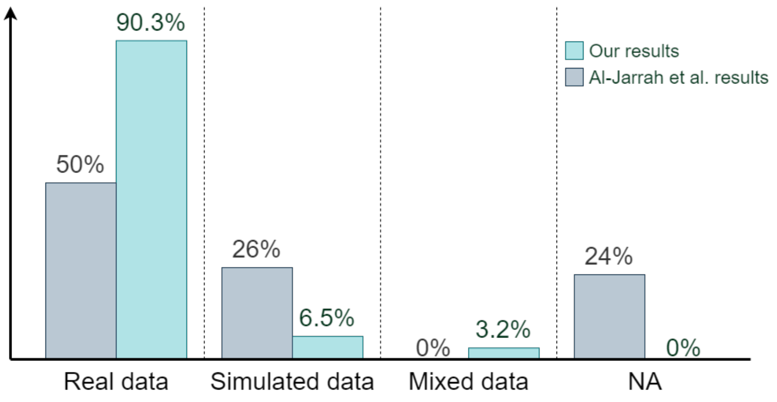

Figure 3.5: Comparison of dataset distribution with [4].

The feature selection process is a decisive step because it can reduce computational costs and improve the detection ability of the IDS. One may think that providing more features to an IDS implies a more effective detection, but in practice this is not the case. Having features not relevant to the target variable or redundant features often influences negatively the detection power of the system, and surely slows down the training and the detection processes; thus it is crucial to select the right subset of features.

It follows the common classification of the feature selection methods:

- **supervised**: these methods use the target variable to discover features not related to it and excluding them.
- **unsupervised**: these methods ignore the target variable and compare the input features with each other in order to find and remove the correlated ones.

Features selection techniques can be further distinguished into:

- **filter methods**: these techniques select features without knowing the detection algorithm. Features are selected on the basis of their scores in correlation with the target value and the features scoring less than a certain threshold are removed.

These methods clearly belong to the supervised class.

- **wrapper methods**: these techniques select features optimizing the learning algorithm. They test many models with different subsets of features and select the best performing one. On the other hand finding an optimal feature subset becomes infeasible when the number of features grows too much, therefore search algorithms are used. Most of these methods are supervised as well.
- **intrinsic methods**: such techniques are the ones included in the learning algorithms, i.e. the model is able to automatically pick the best feature subset. An example of such algorithm is DT.
- **manual**: finally feature selection can also be carried out manually. Indeed when the designers have enough knowledge to understand the problem they are facing, they can choose the most useful features by themselves. Nevertheless it is often difficult to understand deeply the dependency among features, especially when dealing with high-dimensional datasets [4].

Among all the analyzed papers, we found that 25 out of 31 select features manually while 6 use some other methods. In [60] and [38] the PCC method was used; Information Gain (IG) followed by Fast Correlation-Based Filter (FCBF) and Kernel Principal Component Analysis (KPCA) was implemented in [85]; in [63] the authors employed Improved Feature Selection (IFS); CNN was used as feature selection method in [34] and finally in [56] the IDEC method was used.

3.3. Datasets

The evaluation process performed in the reviewed papers is a very important step to understand the capabilities of the proposed IDS. The goal of every designer is to build a solution that can be used in a real-world scenario, thus it is of critical importance the environment used for the evaluation. One of the most important environment characteristic to take into account is the used data: evaluating a model on data that may not represent appropriately a real-world scenario means a not valuable evaluation. For this reason we categorize the datasets used into:

- **real data**: data recorded from real vehicles, usually extracted connecting a recording device to the CAN via the OBD interface.
- **simulated data**: data generated by simulation applications.
- **mixed data**: data is a mix of real data and simulated data.

Obviously the best option is using real datasets because they provide perfect examples of real-world scenarios, while simulated data can exhibit inexistent features or lack existing ones.

As illustrated in Figure 3.5, most of the reviewed papers used real datasets (28 out of 31), while 2 used simulated data ([49] and [64]) and only one employed mixed datasets [28].

Comparing these results with those obtained by Al-Jarrah et al. [4] we can observe a positive trend toward the use of real datasets. This improvement is most likely due to the sharing of data extracted from test vehicles during the last couple of years. We can also notice that most recent works always describe the used dataset, we think that the reason is a deeper knowledge of the problem achieved through time.

3.4. Examined Attack Types

Modern vehicles are vulnerable to different types of cyber attacks with different security levels, from eavesdropping to road users safety threats. Cyber attacks are commonly categorized into active and passive. Passive attacks usually just break the data confidentiality of the system (*e.g.* accessing private CAN frames); while active attacks can cause malfunctions or unintended actions (by inserting, modifying or deleting CAN frames).

To classify the cyber attacks examined in the reviewed papers and to compare our results with [4], we follow their classification method distinguishing the following categories:

- **DoS attack**: the goal of this type of attacks, as the acronym says, is to deny the normal functionality of the CAN-bus sending more messages (not necessarily valid) than the target system handling capabilities. CAN is vulnerable to this type of attacks because of its trivial priority-based arbitration mechanism (see Section 2.1): bursting messages with low CAN-IDs in the network (i.e. *flooding*) is sufficient to achieve a DoS attack. As a consequence, this attack can be launched even by an inexpert attacker as it requires a very limited knowledge of the protocol. The *bus-off attack* [17] is another instance of DoS attack.
- **Message Injection (MI) and Replay attack**: both these types of attacks consist of sending valid frames over the CAN, in particular the replay attack sends old, recorded messages over the network. *Fuzzy* is an injection attack where messages are randomly generated in order to make the system having strange and unpredictable behavior. These attacks are possible because CAN frames are not authenticated, therefore any node connected to the network can send any data.

- **Message Manipulation**: this attack breaks the integrity of transmitted messages by modifying or deleting them. Attackers can perform this attack because CAN is a broadcast channel, so any node can access and manipulate any message.
- **Masquerade attack**: this attack is also known as impersonation attacks, indeed a malicious node can impersonate another ECU. This is possible because there is no way to verify the sender of a message in the CAN. The authors of the reviewed papers refer to this kind of attack also with *spoofing attack*.
- **Malware attacks**: this type of attacks makes use of malwares affecting some ECUs in order to send malicious messages on the network.

We depicted in Figure 3.6 the distribution of attacks examined in the reviewed papers: as we can notice the most considered attack type is the injection one (37 out of 88), similarly to what happens in [4]; then we find DoS attacks close to masquerade ones; 5 manipulation attack examples (in [71], [14], [59] and [28]) and just one malware attack [38]. It can be observed that, differently from [4], we could classify each examined attack within the previous categories. The total number of attacks is much higher than the number of reviewed papers, this is due to the fact that each reasearch usually considered more the one attack type.

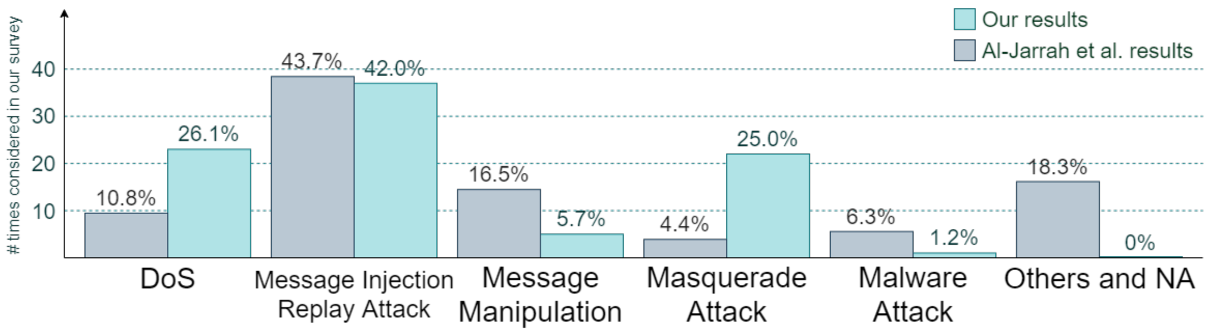

Figure 3.6: Distribution of attack types considered in the reviewed papers and comparison with results of [4].

3.5. Evaluation Metrics

Reading the papers of our survey we found many evaluation metrics, useful to understand the strengths of the proposed models. Here we discuss the performance metrics used for IDSs.

- **Confusion Matrix**: it is a table representing the performance of a classifier. For our purpose the classifiers were binary, thus the confusion matrix has 4 different values. True Positive (TP) is the number of correctly identified intrusions; True Negative (TN) is the number of correctly identified normal messages; False Positive (FP) is the number of incorrectly indentified normal messages as intrusions while False Negative (FN) is the number of incorrectly indentified intrusions as normal messages. A good classifier will have high TP and TN, and low FP and FN.

- **Accuracy**: it is a metric representing a classifier's ability to correctly identify normal messages and intrusions. Its formula is:

$$Acc = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \tag{3.4}$$

  This metric is not appropriate when dealing with imbalanced/skewed datasets.

- **Error Rate**: it is a metric representing a classifier's misclassification rate. Its formula is:

$$ErrorRate = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \tag{3.5}$$

- **True Negative Rate (TNR)**: it is a metric representing the probability of a normal message to be identified as such. Its formula is:

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}. \tag{3.6}$$

- **FPR**: this metric represents the probability of a normal message to be classified as an intrusion. Its formula is:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}}. \tag{3.7}$$

- **False Negative Rate (FNR)**: also known as Miss Rate, this metric represents the probability of an intrusion to be classified as a normal message. Its formula is:

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}}. \tag{3.8}$$

- **Recall**: also known as TPR or Detection Rate (DR), it represents the probability of an intrusion to be identified as such. A recall of 1 means that the classifier correctly detects all intrusions, while a recall of 0 means that no intrusion is identified. Its formula has been defined in Equation (3.2).

- **Precision**: this metric represents the probability of a message identified as intrusion to actually be an intrusion. Its formula has been defined in Equation (3.1).

- **F1-score**: it is the harmonic mean of the precision and recall of the classifier. Its value ranges between 0 and 1, where higher is better. Its formula is:

$$F_1 - score = \frac{2 * \text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}}. \tag{3.9}$$

  Equation (3.3) and Equation (3.9) are equivalent. This metric is commonly used as a single metric to evaluate the effectiveness of a classifier, we used it too in Section 3.1; however it ignores the TN metric which can vary without affecting F1-score. We still used this metric in Section 3.1 to briefly compare the proposed model because authors report it very frequently.

- **Receiver Operating Characteristic (ROC) and Area Under Curve (AUC)**: ROC curve is a visualization of a classifier's performances. It can be drawn plotting FPR against Recall, a perfect classifier will score in the top left corner of a ROC curve while a random classifier will score along the diagonal line of a ROC curve. The AUC quantifies the performance of a classifier. It is the area under the ROC curve, thus it is equal to 1 for a perfect classifier, while it is equal to 0.5 for a random classifier.

- **MCC**: this metric considers every value expressed in the confusion matrix: TP, TN, FP, FN; it is usually seen as an index wrapping them up into a single metric. Its formula is:

$$\text{MCC} = \frac{\text{TP} * \text{TN} - \text{FP} * \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}. \tag{3.10}$$

  Its value ranges between -1, meaning a bad classifier, and +1, which stands for good classifier. As F1-score it is not influenced by dataset imbalances, always providing a sound performance value [59].

However IDSs cannot be evaluated only by their classification ability, indeed they also need to satisfy time requirements. To this end, the Time To Detection (TTD) metric can be found in literature: this metric represents the amount of time elapsed from receiving a CAN frame to classifying it. In literature it is named in different ways (*e.g.* testing time, detection latency). It can be further distinguished in:

- Average TTD: the arithmetic mean of the detection time for each CAN frame.
- WCS-TTD: the maximum detection time recorded among all CAN frames.

An IDS meets time requirements only if the WCS-TTD is within time limits.

3.6. Benchmark Models

Comparing newly designed solutions with benchmark models is a useful practice to understand the performances of the proposed systems. This procedure can show if the new IDS is better (or not) than the literature baseline, becoming the new best solution.

In our survey we found that most of the researches compare their IDSs against at least one benchmark model, this was not the case in Al-Jarrah et al. [4]. In that survey the authors found that just 6 out of 42 reviewed papers compared their solutions with literature's ones, meaning that over the last few years the availability of benchmark models, and the awareness about their importance for this research field, increased significantly (see Figure 3.7).

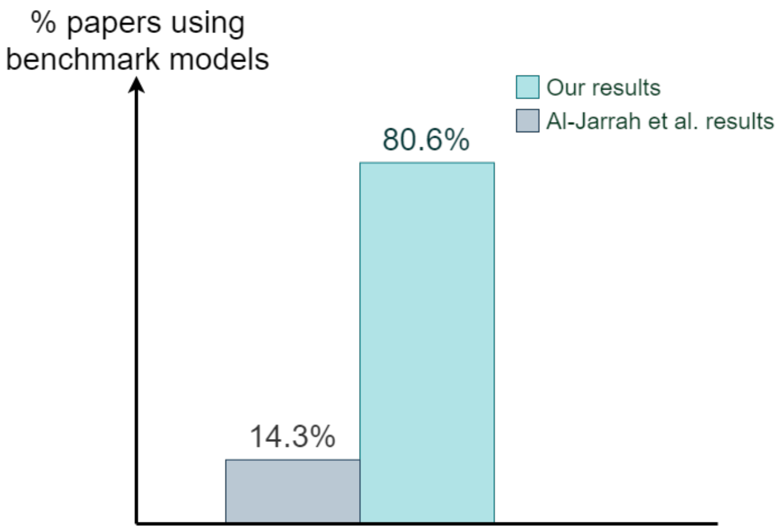

Figure 3.7: Distribution of reviewed papers using benchmark models.

Many types of benchmark models have been used in the reviewed papers, among these you can find ML-based models such as Deep Neural Networks (DNNs), Support Vector Machines (SVMs), K-Nearest Neighbors (K-NN), NB and others; but also traditional algorithms such as interval-based models. Finally some of the most recent researches use models proposed in other papers reviewed by us as benchmarks (*e.g.* Yang et al. [85] compare their solution with *SAIDuCANT* from [62]).

4 | Research Gaps

In this chapter we will analyze the strengths and weaknesses of the solutions found in our survey highlighting the actual gaps from real-world applicability. The Tables 4.1 to 4.3 will help to follow the analysis.

4.1. Detection Techniques and Placement

Figure 4.1 compares the distribution of the detection techniques found by Al-Jarrah et al. [4] with the ones found in the papers reviewed by us: rule-based, T&FA-based, CI&IT-based, hybrid and others. We can observe that not only the CI&IT-based techniques are still prevailing, but also that their predominance has increased. We think that this trend is probably linked to the huge attention gained in the past few years by Artificial Intelligence (AI), in particular ML algorithms, from both academia and industry. Indeed these techniques proved their strength in many applications, intrusion detection included. Nowadays, thanks to this great interest, the development of ML algorithms is accessible to anyone with an easier and easier deployment (*e.g.* TensorFlow framework [2]).

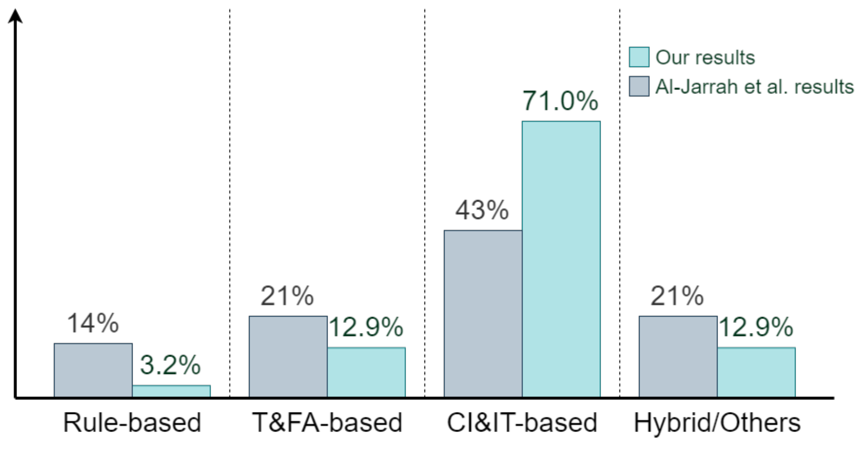

Figure 4.1: Comparison of detection techniques distribution with [4].

Rule-based techniques are the best solutions as long as the problem does not evolve and requires a small rule set, indeed this type of techniques are cheap and computationally

Work	Technique	Features/Feature Selection	Dataset	Attack Type	Performance Metrics	Benchmark Models
[12]	Rule-based	Cyber/NA	ROAD (real data)	MI, Masquerade, Fuzzy	Precision (98.6%), Recall (99.2%), F-score (99.0%)	Mean inter-message time IDS, Fitting Gaussian IDS, Kernel Density Estimation IDS
[86]	Time and Frequency Analysis	Cyber/NA	ORNL (real data)	MI	Acc (100%), FPR (0.7-1.4%)	Interval-based IDSs, GAN-based, DNN-based IDSs
[62]	Time and Frequency Analysis	Cyber/NA	ORNL (real data)	Spoofing, Fuzzy, DoS	Acc (80.3-98.1%), Recall (96.4-99.6%), Precision (80.1-97.7%), F-score (87.5-98.8%), TTD (0-10ms), FPBA (0-1)	Interval-based, Frequency-based IDSs
[27]	Time and Frequency Analysis	Cyber/NA	Hyundai YF Sonata and Kia Soul (real data)	Flooding, Fuzzy, MI, Replay	Acc (99.1-100%), Recall (92.8-99.0%), Time-cost (1-10ms)	Entropy-based, Sequence-based IDSs
[14]	Time and Frequency Analysis	Cyber/NA	3 commercial vehicles (real data)	DoS, Dropping, Fuzzy, Replay, Spoofing	Acc (87.8-99.3%), TPR (83.4-98.9%), FPR ($\leq$ 0.04%), Precision (98.0-99.9%), Mean time-to-detection (3-6ms)	Frequency-based, GAN-based, DCNN-based IDSs, SAIDuCANT
[60]	Computational Intelligence and Information Theory	Cyber/Pearson Correlation Coefficient	Luxgen and Buick (real data)	Spoofing (physical and remote), Bus-off	DR ($\geq$ 96%), FPR ($\leq$ 1%)	SVM
[69]	Computational Intelligence and Information Theory	Cyber/NA	Real data	DoS, Fuzzy, Spoofing	FNR (0.05-0.35%), Error rate (0.03-0-18%), Precision (100%), Recall ($\geq$ 99.9%), F-score ($\geq$ 99.9%), test-time (5ms per batch)	LSTM, ANN, SVM, K-NN, NB, DT
[88]	Computational Intelligence and Information Theory	Cyber/NA	OTIDS [39] (real data)	DoS, Fuzzy	Miss rate (0.49-0.58%), FPR (0.41-0.79%)	NA
[67]	Computational Intelligence and Information Theory	Cyber/NA	Real data	DoS, Fuzzy, Spoofing	Acc (91.3-98.7%), Precision (94.5-97.7%), Recall (83.5-99.8%), F-score (88.6-98.3%)	OCSVM, Deep autoencoder
[55]	Computational Intelligence and Information Theory	Cyber/NA	Hyundai Avante CN7 (real data)	Flooding, Spoofing, Replay, Fuzzy	FNR (0.007-6.276%), F-score (98.97-95.24%)	Bi-direction Markov Method, Uni-directional GPT, Bi-directional LSTM, GAN-based IDS, LSTM-based IDS
[49]	Computational Intelligence and Information Theory	Cyber/NA	Simulated data	Masquerade with Gaslighter	Acc (98.2-99.79%)	Time-interval IDS, ARIMA model, Z-score
[71]	Hybrid	Cyber/NA	ReCAN C-1 (real data)	DoS, Dropping, Fuzzy, Replay	Qualitative results, Time (0.05ms per batch)	Sequence-based IDS
[33]	Other	Cyber/NA	OTIDS [39] (real data)	DoS, Fuzzy, Spoofing, Replay	Acc (94.74-100%), TPR (90.91-100%), FPR (0-100%), TNR (0-100%), FNR (0-7.14%), Time (165.7-258.9μs)	Sequence-based IDS
[5]	Other	Cyber/NA	OTIDS [39] (real data)	DoS, Fuzzy, Impersonation	Acc (98.1%)	Logistic regression, K-NN, DT, CNN, MBA-OCSVM, SVM, GAN

Table 4.1: Summary of Flow-based approaches.

Work	Technique	Features/Feature Selection	Dataset	Attack Type	Performance Metrics	Benchmark Models
[89]	Computational Intelligence and Information Theory	Physical/NA	Real data	Replay	Qualitative results, Latency (2-4ms)	2 undefined models
[30]	Computational Intelligence and Information Theory	Cyber/NA	NAIST (real data)	DoS, Fuzzy, Spoofing	Acc (99.6-100%), Recall (97.6-100%), F-score (97.8-100%), FPR (0-0.2%), FNR (0-2.4%)	Survival Analysis Method
[52]	Computational Intelligence and Information Theory	Cyber/NA	OTIDS [39] (real data)	Spoofing, DoS, Fuzzy	Acc (97.55-100%), TPR (97.6-100%), FPR (0-0.1%), F-score (97.5-100%), Latency (0.002-2ms)	NA
[85]	Computational Intelligence and Information Theory	Cyber/IG followed by FCBF and KPCA	GIDS and CI-CIDS2017 (real data)	DoS, Fuzzy, Spoofing	Acc (99.895-99.999%), TPR (99.806-99.999%), FPR (0.0006-0.084%), F-score (99.895-99.999%), Execution Time (365.3-478.2s)	K-NN, SVM, XYF-K, SAIDuCANT, SSAE, DCNN, LSTM-autoencoder
[64]	Computational Intelligence and Information Theory	Physical/NA	Simulated data	DoS, Fuzzy, Impersonation	Precision (94.5-98.3%), Recall (93-97.8%), F-score (94-97.6%)	NA
[48]	Computational Intelligence and Information Theory	Cyber/NA	Hyundai Avante CN7 (real data)	Fuzzy, Spoofing, Flooding	Acc (97.86-98.10%), Precision (89.58-98.32%), Recall (88.45-98.04%), F-score (90.01-97.83%)	NA
[35]	Computational Intelligence and Information Theory	Cyber/NA	Sonata, Soul and Spark (real data)	DoS, Fuzzy, Impersonation	Qualitative results	Time interval IDS, GAN-based IDS, ML-based IDS
[21]	Computational Intelligence and Information Theory	Cyber/NA	Real data	Fuzzy, Spoofing	Acc (99.24-99.85%), Precision (94.79-98.97%), Recall (99.93-100%), F-score (97.33-99.48%), Latency (0.079ms)	GIDS, DCNN, LSTM-based, MTH-IDS
[58]	Computational Intelligence and Information Theory	Cyber/NA	Hyundai YF Sonata (real data)	DoS, Fuzzy	Acc (100%)	SVM, K-NN

Table 4.2: Summary of Payload-based approaches.

Work	Technique	Features/Feature Selection	Dataset	Attack Type	Performance Metrics	Benchmark Models
[38]	Computational Intelligence and Information Theory	Cyber and Physical/Pearson Correlation Coefficient	OTIDS [39] (real data)	Replay, Amplitude-shift	Acc (83.7-87.9%), Precision (81.2-88.8%), Recall (35.1-83.0%)	MLP, SVM, XGBoost, K-NN, RF, NB
[8]	Computational Intelligence and Information Theory	Cyber/NA	Real data	DoS, Spoofing, Fuzzy	Acc (96.93-100%), Precision (84.78-100%), Recall (97.23-100%), F-score (91.49-100%), FNR (0-2.77%)	NA
[63]	Computational Intelligence and Information Theory	Cyber/Improved Feature Selection	Sonata, Soul and Spark (real data)	Flooding, Fuzzy, Malfunction	Acc (93.36-99.99%), Precision (90.01-99.93%), Recall (91.71-99.98%), F-score (90.85-99.95%), Detection time (0.023ms)	TLMD, SLMC
[34]	Computational Intelligence and Information Theory	Cyber/Convolutional Neural Network	OTIDS [39] (real data)	Fuzzy, DoS, Impersonation	F-score (93.79%), Precision (93.69%), Recall (93.91%)	OCSVM, IF, OTIDS, RNN+heuristics, CANTransfer, DCNN, CANintelliIDS
[28]	Computational Intelligence and Information Theory	Cyber/NA	Mixed data	MI, Camouflage, Suspension, Tampering	Precision (93.87-100%), Acc (97.03-100%), Recall (85.31-100%), F-score (90.49-100%)	NA
[56]	Computational Intelligence and Information Theory	Cyber/Improved Deep Embedded Clustering	Mercedes ML350, ISCX, KDDCup-99, WSN_DS (real data)	DoS, Fuzzy	Acc (78.6-80.3%), Precision (72.8-74.7%), Recall (85.2-86.3%), F-score (78.5-80.1%)	Canet
[7]	Computational Intelligence and Information Theory	Cyber/NA	Real data	Spoofing	TPR (90-100%), FPR (0-0.007%), F-score (95-100%)	LSTM
[59]	Hybrid	Cyber/NA	ReCAN C-1 (real data)	MI, Progressive MI, DoS, Dropping, Fuzzy, Replay	TPR (57.12-99.98%), FPR (1.51-1.90%), F-score (60.36-96.84%), MCC (0.6506-0.9608), TTP (0.2113-0.2688ms)	CANnolo, RNN-based IDS

Table 4.3: Summary of Hybrid approaches.

fast. Unfortunately securing CAN protocol seems to be still a problem able to evolve, we can refer to the discovery of *bus-off attack* [17] from a couple of years ago as an example of evolving problem. Furthermore rule-based algorithms require prior knowledge to build the rule set and they become heavy in terms of memory and computation when a large rule set is employed, which may be the case for securing CAN.

T&FA-based techniques require prior knowledge like rule-based ones, in particular the necessary knowledge is related to the system's normal behavior. At the opposite CI&IT-based techniques do not require any prior knowledge because they set no assumptions on the underlying distribution of the data [4]. CI&IT-based algorithms have the best generalization capability, but most of them have high resource requirements.

It is clear that there is no perfect technique, each comes with some weaknesses. However CI&IT-based techniques just require to operate on performing platforms, thus we will describe a possible solution in Chapter 5.

Another important issue is the placement of the IDS. As depicted in Figure 4.2, the detection device can be placed in two different points: (a) on gateways or (b) on ECUs. The former solution allows the IDS to access data transmitted over different networks making it possible to find correlations among them, on the other hand the IDS must sustain high CAN traffic intensity. With the latter solution the IDS has to classify less CAN frames (only those transmitted to the specific ECU) but the detection algorithm has to adapt to the low computational resources available on an ECU.

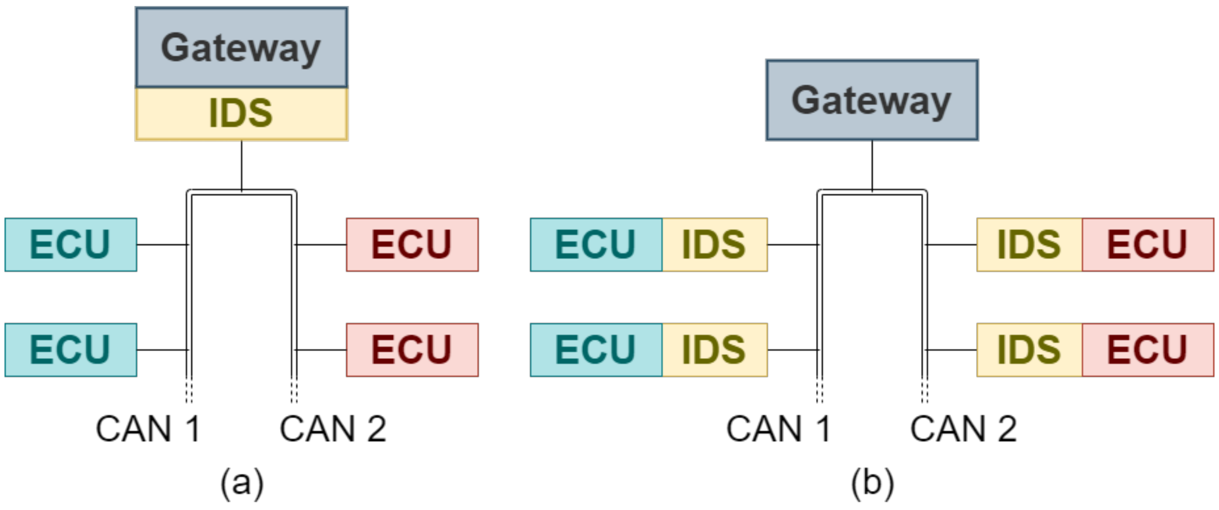

Figure 4.2: IDS placements: (a) IDS is placed on a gateway; (b) IDSs are placed on ECUs.

It is trivial to understand that the IDS placement depends also on the employed detection technique, therefore detection technique and placement should be selected together.

4.2. Features and Feature Selection

Most of the reviewed papers employ only cyber features in order to perform classification (see Figure 3.4). The use of physical features allows to know the current physical state of the vehicle and thus it can improve significantly the IDS performances. However physical features are not always available, indeed their knowledge depends mostly on the availability of the DBC file, which is confidential for vehicle vendors.

As declared by Al-Jarrah et al. [4], IDS research for computer networks has a well-defined feature set, at the opposite there is no specific feature set defined for intra-vehicle IDS research. As already stated in Section 3.2, having more features does not necessarily mean stronger performances, but it surely means longer computation times. Therefore feature selection is a critical step in the design process of an IDS and a potential future research direction.

4.3. Attack Types and Reactive Measures

Most of the reviewed researches have designed IDSs targeting specific attacks (represented in Figure 3.6). However new attack types are likely to show up during a vehicle lifetime, thus another potential research direction can be the deep study of the CAN protocol vulnerabilities in order to define potential unknown attacks. To mitigate this problem we suggest simulating unknown attacks: the simulation can be achieved training the IDS with almost every known attack type and then evaluating the model even on the attack type excluded from the training. In this way the model evaluation considers its performances also on unknown attacks. In order to get a more solid evaluation, this process can be iterated multiple times excluding each time a different attack type.

Another research gap is the countermeasure to dispatch when an attack is detected, indeed IDSs usually just identify intrusions but these are not prevented in any way. Even alerting the vehicle users, as suggested in [4], would be a useless reaction: it is very unlikely that vehicle users have the required knowledge to take the most appropriate action and the alarm may distract the driver leading to a car crash. A more appropriate solution is the IPS proposed in [21]: their method exploits the *bit stuffing* mechanism to transform a malicious CAN frame into an invalid message with the result that other ECUs reject it. However invalidating frames may not always be the most fitting solution, thus establishing a proper set of reactive measures can be the goal of another research direction.

4.4. Performance Metrics and Benchmark Models

Even if most of the reviewed papers use different performance metrics to benchmark the effectiveness of their IDSs, detection latency is a critical metric that researchers rarely report. Comparing with [4], we find that the current situation is slightly better: only 9 out of the 42 papers reviewed in [4] registered this metric, while we found the detection latency reported in 12 out of the 31 reviewed papers, which means an increase of 17%. To understand the importance of this metric we can suppose to have a perfect but very slow intrusion detector: even if it could detect correctly every intrusion, the attacker may already have damaged significantly the targeted system by the detection time. An IDS that does not meet time requirements is almost useless in a real-world scenario. Therefore researchers should consider detection time as a fundamental metric to evaluate an IDS and report it along with the hardware specifications of the platform used for the evaluation. Indeed designers should take into account the reduced computational power available on vehicles, *e.g.* an IDS that can barely keep up with the CAN traffic operating on a high-end modern machine will never sustain the same traffic intensity running on the low-end platforms available on vehicles.

Another important gap for intra-vehicle IDSs research is the lack of a standard benchmark detection model to compare with, as already stated in [4]. This problem still causes inconsistencies in the results reported by different works, which complicates the comparison among the proposed solutions. Therefore the establishment of a baseline model is one more priority research direction because it can ease future developments with standardized benchmarks.

5 | Field Programmable Gate Array for IDSs

In Section 4.1 we observed that among the different categories of detection techniques, the CI&IT-based ones are much more employed than others. Indeed these techniques seem to be the most promising because of their effectiveness and their strong ability for generalization. Moreover CI&IT-based techniques do not require any prior knowledge. The only weakness of these algorithms is related to computation and memory complexity, which means they are resource intensive, thus require powerful platforms in order to be executed with an acceptable throughput. In this chapter we will address this problem with FPGAs.

In the following sections we will analyze FPGAs and their main characteristics in order to understand how they can contribute to the intra-vehicle IDS research field.

5.1. FPGA Overview

An FPGA is an integrated circuit designed to be configured after its fabrication. Conventionally the configuration is specified with a Hardware Description Language (HDL), a similar approach to the one used for Application-Specific Integrated Circuits (ASICs). Unlike ASICs they allow flexible reconfigurable computing like in traditional software as they can be reprogrammed to implement different logic functions.

FPGAs contain a set of Configurable Logic Blocks (CLBs) which can be programmed to carry out specific combinational functions. CLBs may also include memory elements, usually basic flip-flops or data buffers. FPGA architecture also consists of routing channels allowing these blocks to be wired together and I/O ports for external communications.

A basic CLB contains the following smaller components, as depicted in Figure 5.1:

- **Flip-Flop**: a circuit capable of two steady states representing a single bit.
- **Look-Up Table (LUT)**: a collection of gates, storing a predefined list of outputs

for every combination of inputs. LUTs provide a fast way to retrieve the output of a logic operation because possible results are stored and then referenced rather than calculated.

- **Multiplexer**: a circuit that selects and returns one between two or more inputs.

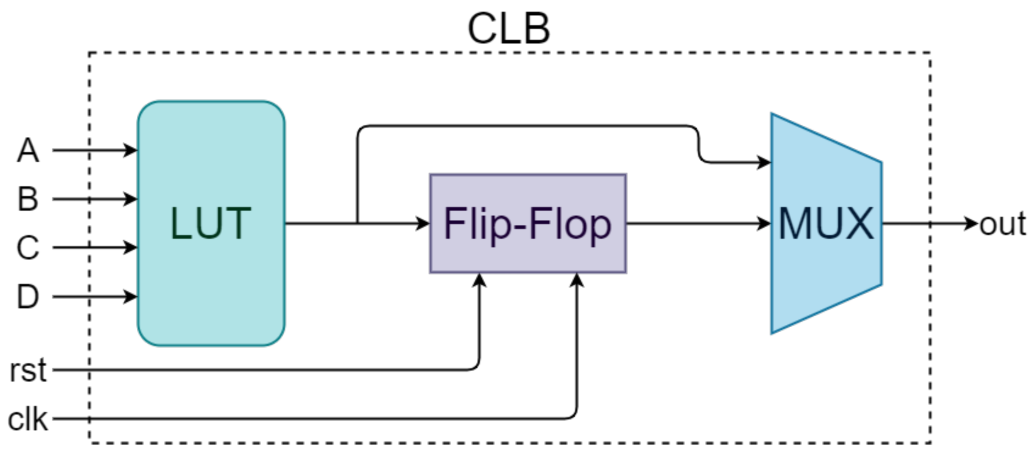

Figure 5.1: Basic CLB structure.

Most programmers are familiar with CPU or GPU -based architectures programming, but developing for FPGAs is significantly different. As a consequence deploying an algorithm on these platforms is a quite difficult task and engineering costs are usually much higher than instruction-based architectures. On the other hand FPGAs provide some advantages which, depending on the targeted problem, can tip the scale in FPGA favor. These advantages are:

- **Latency**: commonly FPGAs have shorter delays than high-end software-based architectures. This result is a direct consequence of the exploitation of specific circuiting rather than relying on general-purpose operating system and communicating over universal bus. Moreover their latency is often deterministic, which is of substantial importance for some use cases such as real-time systems.

- **Energy Efficiency**: FPGAs outperform remarkably CPUs and GPUs regarding energy consumption for specific applications.

- **Connectivity**: any data source, such as a network interface or a sensor, can be directly connected to an FPGA, while CPUs and GPUs use a different approach. For these architectures data sources are connected through standardized buses.

However development complexity and higher costs are not the only drawbacks of FPGAs, indeed these platforms also suffer from lengthy compilation processes: a typical FPGA program compilation usually takes some hours to complete. The compilation process requires a large amount of time and computing power because of its complex optimization algorithm that aims at building the desired circuit configuration with the shortest possible paths.

5.2. FPGA for CAN IDSs

In Section 4.1 we concluded that the only apparent drawback of CI&IT-based detection techniques is their high resource requirements to cope with time restrictions, and in Section 4.4 we explained the importance of detection latency for real-time systems such as intra-vehicle networks. Given the FPGAs advantages over software architectures discussed in Section 5.1, latency in particular, we propose to implement and execute those detection techniques over FPGA architectures.

In order to estimate the real acceleration provided by FPGAs we looked for documents presenting real evaluations of Artificial Neural Networks (ANNs) implemented on a programmable logic board. None of the following papers designed an IDS for CAN: the developed systems aim at different classification problems. Being the IDS a classification problem instance, we can still exploit their results for our estimations.

5.2.1. FPGA Implementation Examples

Nurvitadhi et al. [61] implemented a Gated Recurrent Unit (GRU)-based RNN with a single hidden layer of 256 hidden units. The model was manually developed for the *Altera Stratix 5* and the *Altera Arria 10* programmable logic boards. These platforms are benchmarked against the *Intel Xeon E5-2699v3* CPU clocked at $2.3GHz$ and the *NVIDIA GTX Titan X* GPU. The authors observed that FPGAs have equivalent performances and the same result is obtained for the CPU and the GPU models. The hardware acceleration given by programmable logic is verified by the energy efficiency: the FPGAs $performance/Watt$ metric is 10 times higher than CPU and GPU platforms.

Chang et al. [16] designed a LSTM-based RNN consisting of 2 hidden layers of 128 hidden units each. This model was manually implemented on the *Xilinx Zynq-7000* board. The performances were evaluated with respect to the following platforms:

- the board's CPU (dual-core processor clocked at $667MHz$),
- Tegra K1 CPU (quad-core processor clocked at $2.3GHz$),
- Tegra K1 GPU (Kepler with 192 cores),

- Odroid XU4 Cortex-A15 (quad-core processor clocked at $2GHz$),
- Odroid XU4 Cortex-A7 (quad-core processor clocked at $1.4GHz$).

Because of the quantization process, the model executed on the FPGA suffered from a small accuracy degradation. The best platform among the benchmark models was the *Tegra K1* CPU; with respect to this system, the *Zynq-7000* had 8 times better energy efficiency ($\frac{Mops}{s}/Watt$) and registered half the execution time.

Another LSTM-based RNN model was built by Guan et al. [25]. Their model was structured differently from [16], the hidden layers were 3 and the number of hidden units per layer was 250. The authors implemented the ANN on the *Xilinx VC707* board manually and compared the obtained results with two other platforms: the *Intel Xeon E5-2430* CPU clocked at $2.2GHz$ and the FPGA proposed in [16]. The comparison shows that the execution time of the proposed design is 20 times shorter than the one obtained with the CPU running in single-thread, and 5 times shorter than the one obtained with the CPU running in 16 threads. Finally the design proposed in this paper is 15 times more energy efficient than the one proposed in [16] and almost 2 orders of magnitude more efficient than the CPU-based one.

Wang and Gu [82] implemented the You Only Look Once (YOLO)v3 network, a CNN model, on the *Xilinx ZCU104* MultiProcessor System on a Chip (MPSoC). The implementation process was supported by the use of the *Xilinx*'s recent product, *Vitis AI* [1]: a development platform for AI inference on *Xilinx* hardware platforms (see Figure 5.2). The evaluation process was carried out on the FPGA platform and compared with a *NVIDIA GeForce GTX1080* GPU. The *Xilinx ZCU104* reaches 2.4 times the Frames Per Second (FPS) obtained by the GPU and the energy efficiency (FPS/$Watt$) is 13 times higher.

Tsantikidou et al. [78] built a ANN consisting of 6 layers: 3 dense layers, then 2 LSTM layers and finally another dense layer. The number of hidden units is 64 for every layer but the last one, which has only 6 units. The model was manually implemented on a *Xilinx Alveo U200* board and its performances were compared with the ones obtained with an *Intel Xeon E5-2620 v4* CPU running in single-thread. The execution time on the FPGA system was 39 times shorter than the one of the CPU and the energy efficiency (GFLOPS/$Watt$) registered a 67 times increase on the hardware platform.

Hussein et al. [31] designed a CNN-based autoencoder model where the encoder module is structured by the following 6 layers:

1. First convolutional layer ($16 * 3 * 3$),

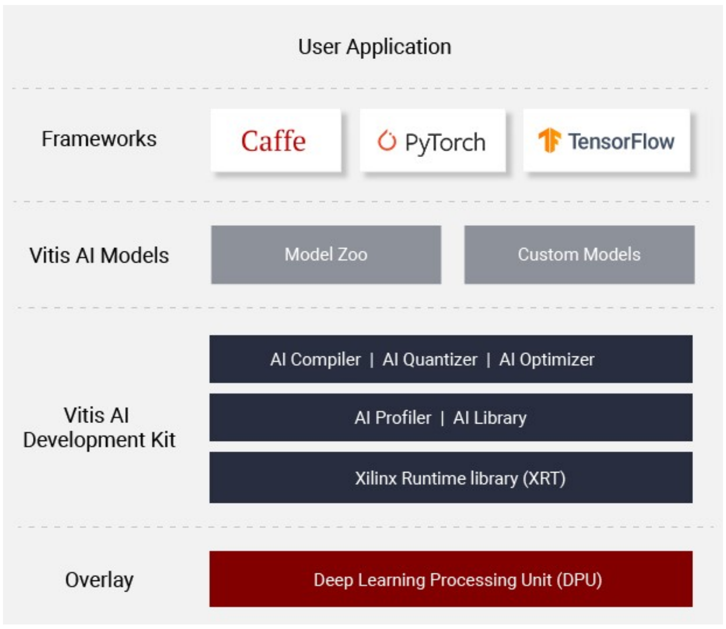

Figure 5.2: *Vitis AI* overview from [1].

2. Second convolutional layer $(8 * 3 * 3)$,
3. First downsampling layer,
4. Third convolutional layer $(4 * 3 * 3)$,
5. Second downsampling layer,
6. Fourth convolutional layer $(1 * 3 * 3)$.

The decoder structure coincides to the specular of the encoder one. The autoencoder was implemented on the *Xilinx ZCU104* MPSoC using the Application Programming Interfaces (APIs) of the *Vitis AI* tool. The authors used a platform powered by an *Intel Core i7-4790* clocked at 3.6GHz with 16GB of RAM to benchmark the performances of the FPGA. The evaluation resulted in an FPGA with 5.5 times faster throughput and a 39.8 times better energy efficiency ($J/Image$) than the CPU platform.

The last reviewed FPGA implementation example is the one proposed by Trabelsi Ajili and Hara-Azumi [77]. The ANN implemented in programmable logic is called *DeepSense*, a multimodal framework (i.e. different types of real-time sensing data are processed). *DeepSense* is a CNN and RNN -based network composed by 8 layers: 3 individual convolutional layers, 3 merge convolutional layers and 2 GRU-based RNN layers. As in [82] and [31], also the authors of this document make use of the *Vitis AI* APIs to develop the model for 2 FPGAs: *Avnet Ultra96-V2* and *Xilinx ZCU102*. These boards are evaluated and compared with the *Nexus 5* smartphone, powered by a quad-core processor clocked at 2.3GHz, and the *Raspberry Pi 3 model B*, powered by a quad-core processor clocked at 1.2GHz. Different configurations were evaluated for the boards, the best with respect to latency is the *Xilinx ZCU102* (*Low-Power Mode* is indifferent) while the best platform for energy efficiency is the *Avnet Ultra96-V2* with low RAM usage, high Digital Signal Processor (DSP) usage[1] and *Low-Power Mode* on. We report the performance increase given by the *Xilinx ZCU102* because our primary concern is latency. This platform reported a 2.9 times shorter latency than *Nexus 5* and a 4.4 times better energy efficiency ($mJ/inference$) than the baseline.

5.2.2. Examples Analysis

The results obtained from the FPGA implementation examples of the previous section are quite variable: the hardware acceleration translates into a performance increase from 2 times to almost 2 orders of magnitude. This large variability is the consequence of different reasons, first of all the distinct classification problems addressed by the documents. The combination of the resources used for benchmarking also affects the comparisons: *e.g.* comparing a high-end CPU against a low-end FPGA will result into a poor performance enhancement.

Regarding latency, we noticed that the ANN models which were implemented in programmable logic employing *Vitis AI* APIs seem to suffer from milder performance boosts. Indeed latency speed up in [82], [31] and [77] is restricted between 2.4× and 5.5×; while in the other documents the latency registers even 39 times shorter values. We suppose this is due to the automatic model optimization performed by *Vitis AI*, which may be less effective than a manual model optimization. However much more samples should be used to prove our hypothesis. Moreover we remind that *Vitis AI* is a recent product and it is continuously developed and improved, thus such an analysis should also take into consideration the used version.

Among the FPGA advantages described in Section 5.1, we can clearly assert that the most noticeable performance improvement, found in these implementation examples, applies to energy efficiency. Even if the evaluation metrics used in the papers to evaluate the energy efficiency are always different, we can observe that its boost is much more evident than the one related to latency.

From the evaluations of the implemented examples, it should be noticed the accuracy degradation phenomenon affecting FPGAs. Indeed, as specified in [16], the implemen-

[1]DSP Usage represents a resource trade-off for DPU's Processing Engine (PE) operations: using more DSPs or LUTs.

tation process for programmable logic boards usually foresees a quantization step. This quantization procedure aims at reducing the model complexity in order to improve the performances on the target platform. Quantization is mandatory in the *Vitis AI* workflow. The accuracy loss should be considered when implementing such techniques on an FPGA.

5.3. FPGA for CANova

In this section we will discuss about the RNN autoencoder-based module used in *CANova* [59] and try to estimate its performance improvement thanks to FPGAs. Among the multiple papers reviewed in Section 3.1 that make use of CI&IT-based detection techniques, we choose to analyze the performance enhancements for [59] because we believe that its modular approach is the most effective one for detecting a wide range of different attack types. Furthermore the document is much more detailed with respect to the others. The RNN autoencoder-based module, although being an effective classifier, is the bottleneck of the *CANova* model; as a matter of fact it is used on all those CAN-IDs for which it does not exist a more appropriate IDS module. Therefore we suggest executing this module on a programmable logic board in order to address its latency issue.

The autoencoder module used in *CANova* is a reduced version of the original *CANnolo* model [42]. The reduced model implemented in *CANova* has the following structure:

- **Encoder**
  - An input layer of dimension $n * k$, where $n = 40$ is the dimension of the window of packets and k is the number of signals.
  - A dense layer composed of 128 units with Exponential Linear Unit (ELU) as activation function.
  - A 20% dropout layer.
  - Two LSTM layers with 64 units each. [59]
- **Decoder**
  - Two LSTM layers with 64 units each.
  - A dense layer composed of 128 units with ELU as activation function.
  - An output dense layer composed of k units with sigmoid activation function. [59]

The main differences with the original *CANnolo* version concern the number of units per layer (it is halved in the reduced model) and the decoder input: in the reduced model the decoder output is not sent back to the decoder input and the encoder output is reversed so that the decoder output is ordered. This last step was not performed in the original *CANnolo* version, indeed its output was in reversed order [42].

Among the different ANN models proposed in the documents analyzed in Section 5.2.1, we think that the one designed by Tsantikidou et al. [78] is the most similar to the RNN autoencoder-based module of *CANova*. Their structures can be compared in Figure 5.3.

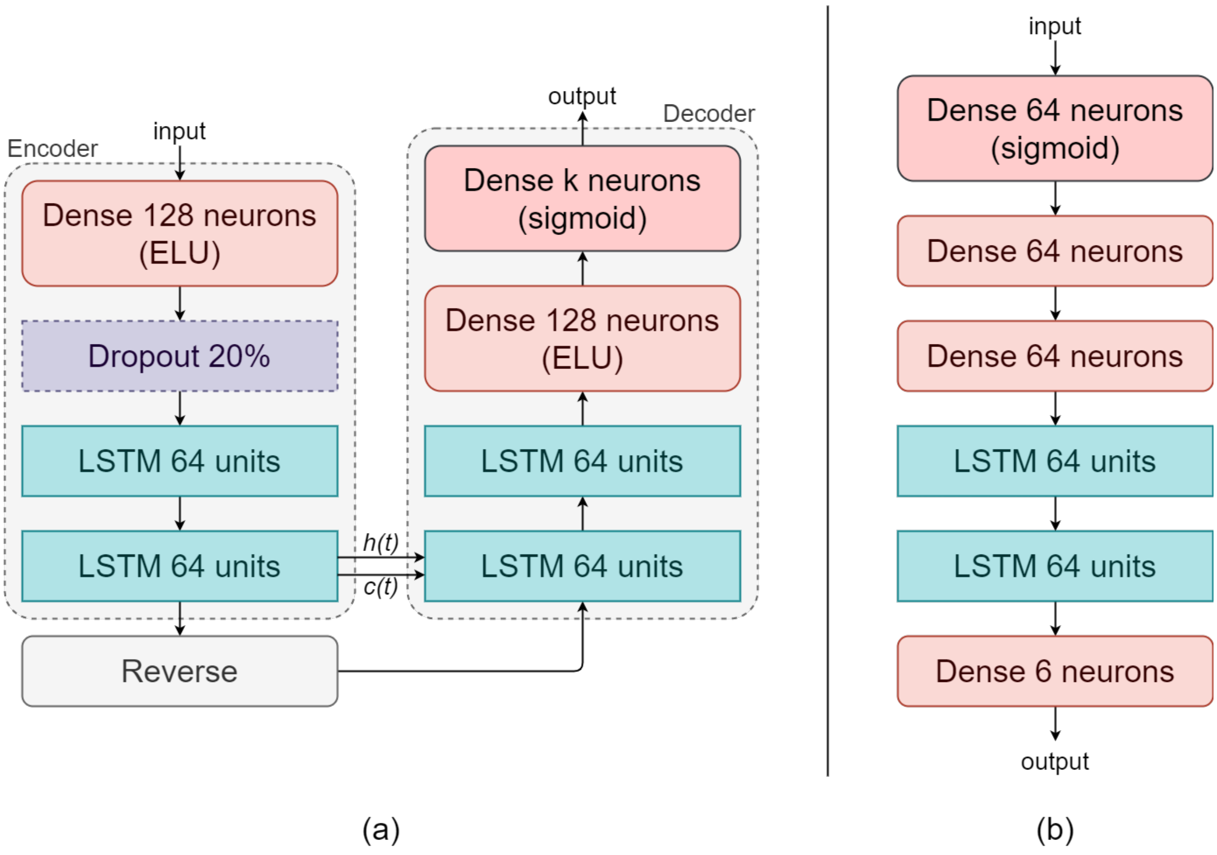

Figure 5.3: Structure comparison of: (a) Reduced *CANnolo* [59] and (b) Tsantikidou et al. [78]'s model.

Considering the fact that the dropout layer of *CANnolo* is used only during the training phase, we can notice that, even if the model proposed in [78] is not designed as an autoencoder, the models have a very close number of layers (7 for *CANnolo* and 6 for [78]) and that the number of units per layer is equal or similar. Moreover the used layer types are the same, dense and LSTM, and they are also arranged in a comparable way: LSTM layers are surrounded by dense ones in both models.

Given these similarities we suppose that the performance improvement provided by the hardware acceleration of FPGAs will be reasonably analogous to the one obtained by Tsantikidou et al. [78]. To support this hypothesis we highlight the fact that the authors of [78] compared a high-end CPU against a high-end FPGA, so that the inequality given by different platform levels does not apply for this particular case. We also remark that the performance enhancement stated in Section 5.2.1 for their design was measured comparing the same optimized C/C++ model on both the platforms; however the RNN autoencoder-based module of *CANova* is not the result of such an optimization process. Indeed the reduced *CANnolo* model was implemented using the Python ML library TensorFlow, thus we can consider it to be the corresponding implementation of the one referred as *Original Python Model* in [78].

For this case the performance boost given by the FPGA reported in [78] is much more remarkable. The comparison between the original Python model and the optimized C/C++ model resulted in an outstanding 152 times shorter latency and 300 times higher energy efficiency. On the other hand the optimized model suffered from an accuracy degradation: the authors of [78] tried many quantization configurations and chose the best one (accordingly to their goal) evaluating the trade-off between latency and accuracy. In particular the selected quantized model has a 5% lower accuracy than the non-quantized configuration.

Such an accuracy loss is not tolerable in our research field because human safety is involved, thus a more accuracy-oriented trade-off choice should be taken into account. As a consequence the latency reduction will not be as marked as for the outcome of [78], but still much appreciable: *e.g.* we refer to the 3^{rd} quantization configuration presented in Tables 1-2 of [78] which has a 2% lower accuracy than the non-quantized one, but still a comparable latency.

The conclusion of the analysis above is the following: given that the TTP of the RNN autoencoder-based module of *CANova* is slightly higher than the average inter-arrival time of CAN frames in the dataset considered in [59], the latency enhancement provided by executing the module on an FPGA would be enough to make the model suitable for real-time detection, and possibly even for more time-stringent scenarios than the one examined in [59].

6 | Conclusions

In this thesis we have given an overview of the CAN protocol currently used on modern vehicles that allows ECUs to share information with one another, then we showed the security risks related to this protocol as it lacks of security measures. This is the starting point of our survey, which aims at the exploration of the state-of-the-art techniques from the research field addressing the just mentioned security threats, renovating Al-Jarrah et al. [4] work.

We recognized some advancements in the research, like the more frequent use of real datasets for testing, and the increased awareness about detection latency that is leading more and more researchers to evaluate their proposed IDSs with this metric. Another significant improvement obtained since the survey of Al-Jarrah et al. [4] is that nowadays the majority of the papers (according to our review) benchmark their solutions against other IDSs taken from literature, this is clearly represented in Figure 3.7.

Unfortunately there is still a lack of conventional baseline models to compare with, as well as standardized test suites that would simplify the comprehension of the proposed model's quality. Moreover the number of researches reporting a critical metric such as detection latency is not big enough yet. For what concerns our survey, it was evaluated only in less than half of the reviewed papers. Furthermore time requirements are very changing from a document to another, making it harder to understand which IDS would actually operate in a real-world environment.

Being latency a major issue for most of the reviewed IDSs, we analyzed FPGA platforms, their benefits and flaws. Then we suggested using them in order to address the real-time requirements of traffic classification. We investigated the supposed speed up provided by such platforms and we supported our theoretical results with the assist of actually implemented instances taken from literature. Our analysis leads to a realistic applicability of *CANova* in a real-world scenario as long as its RNN autoencoder-based module (the bottleneck) is deployed on an FPGA.

Therefore we can conclude that the application of programmable logic boards is a possible solution for addressing timing issues, indeed the hardware acceleration provided by FPGAs seems to be enough to make many models (struggling with latency) able to execute complying with real-world CAN traffic intensity.

Bibliography

[1] Vitis ai, 2020. URL `https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html`. Software available from github.com/Xilinx/Vitis-AI.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[3] S. Abbott-McCune and L. A. Shay. Intrusion prevention system of automotive network can bus. In *2016 IEEE International Carnahan Conference on Security Technology (ICCST)*, pages 1–8, 2016. doi: 10.1109/CCST.2016.7815711.

[4] O. Y. Al-Jarrah, C. Maple, M. Dianati, D. Oxtoby, and A. Mouzakitis. Intrusion detection systems for intra-vehicle networks: A review. *IEEE Access*, 7:21266–21289, 2019. doi: 10.1109/ACCESS.2019.2894183.

[5] I. Aliyu, M. C. Feliciano, S. Van Engelenburg, D. O. Kim, and C. G. Lim. A blockchain-based federated forest for sdn-enabled in-vehicle network intrusion detection system. *IEEE Access*, 9:102593–102608, 2021. doi: 10.1109/ACCESS.2021.3094365.

[6] O. Avatefipour. Physical-fingerprinting of electronic control unit (ecu) based on machine learning algorithm for in-vehicle network communication protocol “can-bus”. 2017.

[7] P. Balaji and M. Ghaderi. Neurocan: Contextual anomaly detection in controller area networks. In *2021 IEEE International Smart Cities Conference (ISC2)*, pages 1–7, 2021. doi: 10.1109/ISC253183.2021.9562830.

[8] V. S. Barletta, D. Caivano, A. Nannavecchia, and M. Scalera. Intrusion detection

for in-vehicle communication networks: An unsupervised kohonen som approach. *Future Internet*, 12(7), 2020. ISSN 1999-5903. doi: 10.3390/fi12070119. URL `https://www.mdpi.com/1999-5903/12/7/119`.

[9] O. Berlin, A. Held, M. Matousek, and F. Kargl. Poster: Anomaly-based misbehaviour detection in connected car backends. In *2016 IEEE Vehicular Networking Conference (VNC)*, pages 1–2, 2016. doi: 10.1109/VNC.2016.7835978.

[10] A. Bezemskij, G. Loukas, R. J. Anthony, and D. Gan. Behaviour-based anomaly detection of cyber-physical attacks on a robotic vehicle. In *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*, pages 61–68, 2016. doi: 10.1109/IUCC-CSS.2016.017.

[11] A. Bezemskij, G. Loukas, D. Gan, and R. J. Anthony. Detecting cyber-physical threats in an autonomous robotic vehicle using bayesian networks. In *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 98–103, 2017. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.20.

[12] D. H. Blevins, P. Moriano, R. A. Bridges, M. E. Verma, M. D. Iannacone, and S. C. Hollifield. Time-based can intrusion detection benchmark. *Proceedings Third International Workshop on Automotive and Autonomous Vehicle Security*, 2021. doi: 10.14722/autosec.2021.23013. URL `http://dx.doi.org/10.14722/autosec.2021.23013`.

[13] A. Boudguiga, W. Klaudel, A. Boulanger, and P. Chiron. A simple intrusion detection method for controller area network. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–7, 2016. doi: 10.1109/ICC.2016.7511098.

[14] M. Bozdal, M. Samie, and I. K. Jennions. Winds: A wavelet-based intrusion detection system for controller area network (can). *IEEE Access*, 9:58621–58633, 2021. doi: 10.1109/ACCESS.2021.3073057.

[15] T. C. Cañones. Benchmarking framework for intrusion detection systems in controller area networks. Master's thesis, Politecnico di Milano, 2021.

[16] A. X. M. Chang, B. Martini, and E. Culurciello. Recurrent neural networks hardware implementation on FPGA. *CoRR*, abs/1511.05552, 2015. URL `http://arxiv.org/abs/1511.05552`.

[17] K.-T. Cho and K. G. Shin. Error handling of in-vehicle networks makes them vulnerable. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1044–1055, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978302. URL `https://doi.org/10.1145/2976749.2978302`.

[18] K.-T. Cho and K. G. Shin. Fingerprinting electronic control units for vehicle intrusion detection. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, pages 911–927, USA, 2016. USENIX Association. ISBN 9781931971324.

[19] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee. Voltageids: Low-level communication characteristics for automotive intrusion detection system. *IEEE Transactions on Information Forensics and Security*, 13(8):2114–2129, 2018. doi: 10.1109/TIFS.2018.2812149.

[20] Cia. Can data link layers in some detail. URL `https://www.can-cia.org/can-knowledge/can/can-data-link-layers/`.

[21] P. Freitas De Araujo-Filho, A. J. Pinheiro, G. Kaddoum, D. R. Campelo, and F. L. Soares. An efficient intrusion prevention system for can: Hindering cyber-attacks with a low-cost platform. *IEEE Access*, 9:166855–166869, 2021. doi: 10.1109/ACCESS.2021.3136147.

[22] W. Fu, X. Xin, P. Guo, and Z. Zhou. A practical intrusion detection system for internet of vehicles. *China Communications*, 13(10):263–275, 2016. doi: 10.1109/CC.2016.7733050.

[23] A. Ganesan, J. Rao, and K. G. Shin. Exploiting consistency among heterogeneous sensors for vehicle anomaly detection. 2017.

[24] M. Gmiden, M. H. Gmiden, and H. Trabelsi. An intrusion detection method for securing in-vehicle can bus. In *2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pages 176–180, 2016. doi: 10.1109/STA.2016.7952095.

[25] Y. Guan, Z. Yuan, G. Sun, and J. Cong. Fpga-based accelerator for long short-term memory recurrent neural networks. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 629–634, 2017. doi: 10.1109/ASPDAC.2017.7858394.

[26] R. E. Haas, D. P. F. Möller, P. Bansal, R. Ghosh, and S. S. Bhat. Intrusion detection

in connected cars. In *2017 IEEE International Conference on Electro Information Technology (EIT)*, pages 516–519, 2017. doi: 10.1109/EIT.2017.8053416.

[27] M. L. Han, B. I. Kwak, and H. K. Kim. Event-triggered interval-based anomaly detection and attack identification methods for an in-vehicle network. *IEEE Transactions on Information Forensics and Security*, 16:2941–2956, 2021. doi: 10.1109/TIFS.2021.3069171.

[28] X. He, Z. Yang, and Y. Huang. A vehicle intrusion detection system based on time interval and data fields. In X. Sun, X. Zhang, Z. Xia, and E. Bertino, editors, *Artificial Intelligence and Security*, pages 538–549, Cham, 2021. Springer International Publishing. ISBN 978-3-030-78612-0.

[29] T. Hoppe, S. Kiltz, and J. Dittmann. Applying intrusion detection to automotive it-early insights and remaining challenges. *Journal of Information Assurance and Security (JIAS)*, 4:226–235, 01 2009.

[30] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi. Lstm-based intrusion detection system for in-vehicle can bus communications. *IEEE Access*, 8: 185489–185502, 2020. doi: 10.1109/ACCESS.2020.3029307.

[31] A. S. Hussein, A. Anwar, Y. Fahmy, H. Mostafa, K. N. Salama, and M. Kafafy. Implementation of a dpu-based intelligent thermal imaging hardware accelerator on fpga. *Electronics*, 11(1), 2022. ISSN 2079-9292. doi: 10.3390/electronics11010105. URL `https://www.mdpi.com/2079-9292/11/1/105`.

[32] T. Instruments. Introductionto the controllerareanetwork(can). 2002.

[33] R. Islam, R. U. D. Refat, S. M. Yerram, and H. Malik. Graph-based intrusion detection system for controller area networks. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–10, 2020. doi: 10.1109/TITS.2020.3025685.

[34] A. R. Javed, S. u. Rehman, M. U. Khan, M. Alazab, and T. R. G. Canintelliids: Detecting in-vehicle intrusion attacks on a controller area network using cnn and attention-based gru. *IEEE Transactions on Network Science and Engineering*, 8(2): 1456–1466, 2021. doi: 10.1109/TNSE.2021.3059881.

[35] L. Kang and H. Shen. A transfer learning based abnormal can bus message detection system. In *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, pages 545–553, 2021. doi: 10.1109/MASS52906.2021.00073.

[36] M.-J. Kang and J.-W. Kang. 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, 2016. doi: 10.1109/VTCSpring.2016.7504089.

[37] M.-J. Kang and J.-W. Kang. Intrusion detection system using deep neural network for in-vehicle network security. *PLOS ONE*, 11(6):1–17, 06 2016. doi: 10.1371/journal.pone.0155781. URL `https://doi.org/10.1371/journal.pone.0155781`.

[38] Z. Khan, M. Chowdhury, M. Islam, C.-Y. Huang, and M. Rahman. Long short-term memory neural network-based attack detection model for in-vehicle network security. *IEEE Sensors Letters*, 4(6):1–4, 2020. doi: 10.1109/LSENS.2020.2993522.

[39] H. Lee, S. H. Jeong, and H. K. Kim. Otids: A novel intrusion detection system for in-vehicle network by using remote frame. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, volume 00, pages 57–5709, 8 2017. doi: 10.1109/PST.2017.00017. URL `doi.ieeecomputersociety.org/10.1109/PST.2017.00017`.

[40] M. Levi, Y. Allouche, and A. Kontorovich. Advanced analytics for connected car cybersecurity. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pages 1–7, 2018. doi: 10.1109/VTCSpring.2018.8417690.

[41] C. Ling and D. Feng. An algorithm for detection of malicious messages on can buses. 11 2012. doi: 10.2991/citcs.2012.161.

[42] S. Longari, D. H. Nova Valcarcel, M. Zago, M. Carminati, and S. Zanero. Cannolo: An anomaly detection system based on lstm autoencoders for controller area network. *IEEE Transactions on Network and Service Management*, 18(2):1913–1924, 2021. doi: 10.1109/TNSM.2020.3038991.

[43] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D. Gan. Cloud-based cyber-physical intrusion detection for vehicles using deep learning. *IEEE Access*, 6: 3491–3508, 2018. doi: 10.1109/ACCESS.2017.2782159.

[44] M. Marchetti and D. Stabili. Anomaly detection of can bus messages through analysis of id sequences. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1577–1583, 2017. doi: 10.1109/IVS.2017.7995934.

[45] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni. Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–6, 2016. doi: 10.1109/RTSI.2016.7740627.

[46] M. Markovitz and A. Wool. Field classification, modeling and anomaly detection

in unknown can bus networks. *Vehicular Communications*, 9:43–52, 2017. ISSN 2214-2096. doi: https://doi.org/10.1016/j.vehcom.2017.02.005. URL `https://www.sciencedirect.com/science/article/pii/S2214209616300869`.

[47] F. Martinelli, F. Mercaldo, V. Nardone, and A. Santone. Car hacking identification through fuzzy logic algorithms. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–7, 2017. doi: 10.1109/FUZZ-IEEE.2017.8015464.

[48] S. T. Mehedi, A. Anwar, Z. Rahman, and K. Ahmed. Deep transfer learning based intrusion detection system for electric vehicular networks. *Sensors*, 21(14), 2021. ISSN 1424-8220. doi: 10.3390/s21144736. URL `https://www.mdpi.com/1424-8220/21/14/4736`.

[49] J. Mehta and B. Meyer. Dt-ds: Can intrusion detection with decision tree ensembles. URL `https://escholarship.mcgill.ca/concern/theses/f4752n789`.

[50] C. Miller and C. Valasek. Adventures in automotive networks and control units. *Def Con*, 21(260-264):15–31, 2013.

[51] C. Miller and C. Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015(S 91), 2015.

[52] O. Minawi, J. Whelan, A. Almehmadi, and K. El-Khatib. Machine learning-based intrusion detection system for controller area networks. In *Proceedings of the 10th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, DIVANet '20, pages 41–47, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450381215. doi: 10.1145/3416014.3424581. URL `https://doi.org/10.1145/3416014.3424581`.

[53] M. Müter and N. Asaj. Entropy-based anomaly detection for in-vehicle networks. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 1110–1115, 2011. doi: 10.1109/IVS.2011.5940552.

[54] M. Müter, A. Groll, and F. C. Freiling. A structured approach to anomaly detection for in-vehicle networks. In *2010 Sixth International Conference on Information Assurance and Security*, pages 92–98, 2010. doi: 10.1109/ISIAS.2010.5604050.

[55] M. Nam, S. Park, and D. S. Kim. Intrusion detection method using bi-directional gpt for in-vehicle controller area networks. *IEEE Access*, 9:124931–124944, 2021. doi: 10.1109/ACCESS.2021.3110524.

[56] H. Narasimhan, V. Ravi, and N. Mohammad. Unsupervised deep learning approach

for in-vehicle intrusion detection system. *IEEE Consumer Electronics Magazine*, pages 1–1, 2021. doi: 10.1109/MCE.2021.3116923.

[57] S. N. Narayanan, S. Mittal, and A. Joshi. Using data analytics to detect anomalous states in vehicles, 2015. URL `https://arxiv.org/abs/1512.08048`.

[58] I. Nazakat and K. Khurshid. Intrusion detection system for in-vehicular communication. In *2019 15th International Conference on Emerging Technologies (ICET)*, pages 1–6, 2019. doi: 10.1109/ICET48972.2019.8994327.

[59] A. Nichelini and C. A. Pozzoli. Canova, a classification-based modular intrusion detection system for can. Master's thesis, Politecnico di Milano, 2021.

[60] J. Ning, J. Wang, J. Liu, and N. Kato. Attacker identification and intrusion detection for in-vehicle networks. *IEEE Communications Letters*, 23(11):1927–1930, 2019. doi: 10.1109/LCOMM.2019.2937097.

[61] E. Nurvitadhi, J. Sim, D. Sheffield, A. Mishra, S. Krishnan, and D. Marr. Accelerating recurrent neural networks in analytics servers: Comparison of fpga, cpu, gpu, and asic. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4, 2016. doi: 10.1109/FPL.2016.7577314.

[62] H. Olufowobi, C. Young, J. Zambreno, and G. Bloom. Saiducant: Specification-based automotive intrusion detection using controller area network (can) timing. *IEEE Transactions on Vehicular Technology*, 69(2):1484–1494, 2020. doi: 10.1109/TVT.2019.2961344.

[63] S. Park and J.-Y. Choi. Hierarchical anomaly detection model for in-vehicle networks using machine learning algorithms. *Sensors*, 20(14), 2020. ISSN 1424-8220. doi: 10.3390/s20143934. URL `https://www.mdpi.com/1424-8220/20/14/3934`.

[64] F. Pascale, E. A. Adinolfi, S. Coppola, and E. Santonicola. Cybersecurity in automotive: An intrusion detection system in connected vehicles. *Electronics*, 10 (15), 2021. ISSN 2079-9292. doi: 10.3390/electronics10151765. URL `https://www.mdpi.com/2079-9292/10/15/1765`.

[65] R. Rieke, M. Seidemann, E. K. Talla, D. Zelle, and B. Seeger. Behavior analysis for safety and security in automotive systems. In *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 381–385, 2017. doi: 10.1109/PDP.2017.67.

[66] P. Sharma and D. P. F. Möller. Protecting ecus and vehicles internal networks.

In *2018 IEEE International Conference on Electro/Information Technology (EIT)*, pages 0465–0470, 2018. doi: 10.1109/EIT.2018.8500295.

[67] H. M. Song and H. K. Kim. Self-supervised anomaly detection for in-vehicle network using noised pseudo normal data. *IEEE Transactions on Vehicular Technology*, 70 (2):1098–1108, 2021. doi: 10.1109/TVT.2021.3051026.

[68] H. M. Song, H. R. Kim, and H. K. Kim. Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network. In *2016 International Conference on Information Networking (ICOIN)*, pages 63–68, 2016. doi: 10.1109/ICOIN.2016.7427089.

[69] H. M. Song, J. Woo, and H. K. Kim. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, 21:100198, 2020. ISSN 2214-2096. doi: https://doi.org/10.1016/j.vehcom.2019.100198. URL `https://www.sciencedirect.com/science/article/pii/S2214209619302451`.

[70] D. Stabili, M. Marchetti, and M. Colajanni. Detecting attacks to internal vehicle networks through hamming distance. In *2017 AEIT International Annual Conference*, pages 1–6, 2017. doi: 10.23919/AEIT.2017.8240550.

[71] J. Sunny, S. Sankaran, and V. Saraswat. A hybrid approach for fast anomaly detection in controller area networks. In *2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6, 2020. doi: 10.1109/ANTS50601.2020.9342791.

[72] C. P. Szydlowski. Can specification 2.0: Protocol and implementations. In *Future Transportation Technology Conference and Exposition*. SAE International, 8 1992. doi: https://doi.org/10.4271/921603. URL `https://doi.org/10.4271/921603`.

[73] A. Taylor, N. Japkowicz, and S. Leblanc. Frequency-based anomaly detection for the automotive can bus. In *2015 World Congress on Industrial Control Systems Security (WCICSS)*, pages 45–49, 2015. doi: 10.1109/WCICSS.2015.7420322.

[74] A. Taylor, S. Leblanc, and N. Japkowicz. Anomaly detection in automobile control network data with long short-term memory networks. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 130–139, 2016. doi: 10.1109/DSAA.2016.20.

[75] A. Theissler. Anomaly detection in recordings from in-vehicle networks. 09 2014.

[76] A. Theissler. Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection. *Knowledge-Based Systems*, 123:163–173, 2017.

ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2017.02.023. URL `https://www.sciencedirect.com/science/article/pii/S0950705117301077`.

[77] M. Trabelsi Ajili and Y. Hara-Azumi. Multimodal neural network acceleration on a hybrid cpu-fpga architecture: A case study. *IEEE Access*, 10:9603–9617, 2022. doi: 10.1109/ACCESS.2022.3144977.

[78] K. Tsantikidou, N. Tampouratzis, and I. Papaefstathiou. A novel fpga-based intent recognition system utilizing deep recurrent neural networks. *Electronics*, 10(20): 2495, 10 2021. ISSN 2079-9292. doi: 10.3390/electronics10202495. URL `http://dx.doi.org/10.3390/electronics10202495`.

[79] D. K. Vasistha, 2017. URL `https://hdl.handle.net/1969.1/165769`.

[80] T. P. Vuong, G. Loukas, and D. Gan. Performance evaluation of cyber-physical intrusion detection on a robotic vehicle. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 2106–2113, 2015. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.313.

[81] C. Wang, Z. Zhao, L. Gong, L. Zhu, Z. Liu, and X. Cheng. A distributed anomaly detection system for in-vehicle network using htm. *IEEE Access*, 6:9091–9098, 2018. doi: 10.1109/ACCESS.2018.2799210.

[82] J. Wang and S. Gu. Fpga implementation of object detection accelerator based on vitis-ai. In *2021 11th International Conference on Information Science and Technology (ICIST)*, pages 571–577, 2021. doi: 10.1109/ICIST52614.2021.9440554.

[83] A. R. Wasicek, M. D. Pesé, A. Weimerskirch, Y. Burakova, and K. Singh. Context-aware intrusion detection in automotive control system, 2017.

[84] M. Weber, S. Klug, B. Zimmer, and E. Sax. Embedded hybrid anomaly detection for automotive can communication. 02 2018.

[85] L. Yang, A. Moubayed, and A. Shami. Mth-ids: A multitiered hybrid intrusion detection system for internet of vehicles. *IEEE Internet of Things Journal*, 9(1): 616–632, 2022. doi: 10.1109/JIOT.2021.3084796.

[86] C. Young, H. Olufowobi, G. Bloom, and J. Zambreno. Automotive intrusion detection based on constant can message frequencies across vehicle driving modes. URL `https://dl.acm.org/doi/10.1145/3309171.3309179`.

[87] C. Young, J. Zambreno, and G. Bloom. Towards a fail-operational intrusion detection

system for in-vehicle networks. In *Proceedings of the Workshop on Security and Dependability of Critical Embedded Real-Time Systems (CERTS)*, November 2016.

[88] K.-S. Yu, S.-H. Kim, D.-W. Lim, and Y.-S. Kim. A multiple rényi entropy based intrusion detection system for connected vehicles. *Entropy*, 22(2), 2020. ISSN 1099-4300. doi: 10.3390/e22020186. URL `https://www.mdpi.com/1099-4300/22/2/186`.

[89] J. Zhang, F. Li, H. Zhang, R. Li, and Y. Li. Intrusion detection system using deep learning for in-vehicle security. *Ad Hoc Networks*, 95:101974, 2019. ISSN 1570-8705. doi: https://doi.org/10.1016/j.adhoc.2019.101974. URL `https://www.sciencedirect.com/science/article/pii/S1570870519304354`.

[90] L. Zhang, L. Shi, N. Kaja, and D. Ma. A two-stage deep learning approach for can intrusion detection. 2018.

[91] T. Zhang, H. Antunes, and S. Aggarwal. Defending connected vehicles against malware: Challenges and a solution framework. *IEEE Internet of Things Journal*, 1(1): 10–21, 2014. doi: 10.1109/JIOT.2014.2302386.

A | Appendix A

In this appendix we report the summary tables detailed in [4] in order to allow the readers to compare the papers reviewed by Al-Jarrah et al. with the ones analyzed in our survey.

Work	Technique	Features/Feature Selection	Dataset	Attack Type	Performance Metrics	Benchmark Models
[80]	Rule-based	Cyber and Physical/NA	Real data of 52,215 records of a small robot vehicle	DoS, MI, 2 Malwares	Acc (66.7-85.24%), FPR (5.43-29.60%), FNR (5.74-41.44%), ROC, AUC (0.73-0.97), Latency (about 1 sec)	NA
[22]	Rule-based	Cyber/NA	NA	NA	Throughput (39Gbps), Power consumption (7.5W), Latency ($4\mu s$)	Snort software
[29]	Time and Frequency Analysis	Cyber/NA	Simulated data	MI	NA	NA
[41]	Time and Frequency Analysis	Cyber/NA	Simulated data	NA	NA	NA
[73]	Time and Frequency Analysis	Cyber/NA	Real data of a Ford Explorer 2011	MI, Deletion	ROC, AUC (0.8720-1.0000)	OCSVM
[68]	Time and Frequency Analysis	Cyber/NA	Real data of an anomanyised vehicle	MI	Acc (36-100%)	NA
[87]	Time and Frequency Analysis	Cyber/NA	NA	Replay, MI	NA	NA
[18]	Time and Frequency Analysis	Cyber/NA	Real data of approximately 2.25 million messages	MI, Deletion, Masquerade	ROC, FPR (0.055%), TPR (100%)	NA
[24]	Time and Frequency Analysis	Cyber/NA	NA	MI, Invalid messages	NA	NA
[39]	Time and Frequency Analysis	Cyber/NA	Real data captured from a Kia Soul	DoS, Fuzzy, Masquerade	NA	NA
[6]	Time and Frequency Analysis	Physical/Joint Mutual Information Criterion	A dataset collected from 16 different channels and 4 identical ECUs transmitting the same message	MI	DR for channel (95.2%), DR for ECU (98.3%)	NA
[53]	Computational Intelligence and Information Theory	Cyber/NA	Real data	MI, DoS, Plausibility of Interrelated Events	NA	NA
[45]	Computational Intelligence and Information Theory	Cyber/NA	Real data from 2011 Ford Fiesta	MI, Fuzzy	NA	NA
[40]	Computational Intelligence and Information Theory	Cyber/NA	Simulated data	USB firmware update, Communication with unknown vendor, OTA malicious updates, Malicious application installation	AUC (0.81-0.96), F-score (0.61-0.93)	NA
[65]	Computational Intelligence and Information Theory	Cyber/NA	Real data from Renault Zoe 2016	MI	Throughput and complexity ($> 2000 events/s$)	NA
[83]	Computational Intelligence and Information Theory	Physical/NA	Real data of a 2015 passenger vehicle	Chip tuning (manipulation)	NA	NA
[19]	Computational Intelligence and Information Theory	Cyber/Sequential Forward Selection	Real data from a prototype setup, from Hyundai Sonata 2010, from Kia Soul 2014	Masquerade, Bus-off	F-score (54.24-99.61%), Precision (92-99%), Recall (92-99%), FPR (0%)	NA
[13]	Others/Hybrid	Cyber/NA	NA	Masquerade, DoS, Replay	NA	NA
[44]	Others/Hybrid	Cyber/NA	Real data from an unmodified licensed vehicle	Replay, MI	DR (20-100%)	NA

Table A.1: Summary of Flow-based approaches reviewed in [4].

Work	Technique	Features/Feature Selection	Dataset	Attack Type	Performance Metrics	Benchmark Models
[10]	Rule-based	Cyber and Physical/NA	NA	Replay, MI, Rogue node, Compass manipulation, Broken wheel	ROC, AUC (0.406-1)	NA
[3]	Rule-based	Cyber/NA	Simulated data	Replay, Invalid messages (MI)	NA	NA
[46]	Rule-based	Cyber/NA	Mixed data	MI	Field Classification Distance (10-40.8%), FPR (0.1-2.2%)	NA
[70]	Rule-based	Cyber/NA	Real data from 2011 Ford Fiesta	MI	DR (20-100%)	NA
[75]	Computational Intelligence and Information Theory	Cyber/NA	Real data	Faults	Training Time (20843-63631s), Testing Time (4845-24145s), FN (0.0-10.5/hour), TN (9-45/hour), TNR (42.9-76.9%), Precision (32.3-100%)	NA
[57]	Computational Intelligence and Information Theory	Cyber/NA	Real data from Honda, Toyota and Chevrolet	MI	NA	NA
[36]	Computational Intelligence and Information Theory	Cyber/NA	Simulated data	Field modification, MI	TPR (97.6-99.8%), TNR (93.7-99.9%), Training Time (4.741-11.977s), Testing Time (8.957-8.120ms)	ANN
[37]	Computational Intelligence and Information Theory	Cyber/NA	Simulated data	MI, Manipulation attacks	ROC, DR (99%), FPR (1.6%), Acc(97.8%), Training Time (4.15-10.81s), Testing Time (2.05-3.78ms)	SVM, ANN
[74]	Computational Intelligence and Information Theory	Cyber/NA	Real data from 2012 Subaru Impreza	Interleave, Drop, Discontinuity, Unusual, Reverse	ROC, AUC (0.176471-1), TPR (100%), FPR (0.0010-0.6341%)	NA
[76]	Computational Intelligence and Information Theory	Cyber/NA	Real data from 2002 Renault Twingo	Faults injection	TPR (80-100%), Precision (35.4-100%), F2-score (68.5-83.3%), Diversity (0.093-0.223)	NA
[23]	Computational Intelligence and Information Theory	Cyber/NA	Real data	MI	NA	NA
[47]	Computational Intelligence and Information Theory	Cyber/NA	Real data	DoS, Fuzzy, MI	FPR (0-0.038), Precision (0.963-1), Recall (0.823-1), F-score (0.981-1), AUC (0.986-1)	NA
[26]	Computational Intelligence and Information Theory	Cyber/NA	NA	NA	NA	NA
[66]	Computational Intelligence and Information Theory	Cyber/NA	NA	NA	NA	NA

Table A.2: Summary of Payload-based approaches reviewed in [4]. Part 1

Work	Technique	Features/Feature Selection	Dataset	Attack Type	Performance Metrics	Benchmark Models
[81]	Computational Intelligence and Information Theory	Cyber/NA	Real data from Subaru Impreza	Field modification, Replay	AUC (0.85-1), Recall (0.2-0.8), Precision (0.95-1)	RNN, Hidden Markov Model
[43]	Computational Intelligence and Information Theory	Cyber and Physical/NA	Simulated data	DoS, MI, Malware	Acc (86.9%), Testing Time (1.163-1.704)	Logistic regression, DT, SVM, RF, MLP
[11]	Others/Hybrid	Cyber and Physical/NA	Data of a robot vehicle	MI, Rogue node, Magnetic Interference attacks	ROC, AUC (0.995)	NA

Table A.3: Summary of Payload-based approaches reviewed in [4]. Part 2

Work	Technique	Features/Feature Selection	Dataset	Attack Type	Performance Metrics	Benchmark Models
[54]	Detection sensors	NA	NA	NA	NA	NA
[91]	A cloud-assisted vehicle malware protection framework	Cyber/NA	NA	Malware	NA	NA
[9]	A back-end Security Information and Event Management System	NA	NA	NA	NA	NA
[79]	Cross-correlation-based detector, Timing-based detector, Messages order detector	Cyber/NA	Real data from Honda Civic, Toyota Camry, Kia	MI, Deletion, DoS	Latency (2s), FPR (0-3.5%)	NA
[84]	A hybrid approach of specification-based and ML-based system	Cyber/NA	Simulated data	Limitation of value range, Value freeze, Alternative signal sequence, Peak signal, Signal jump	NA	NA
[90]	A hybrid approach of rule-based and ML-based system	Cyber/Forward Feature Seletion	Real data	MI, Spoofing, Replay, Deletion	DR (99.91-99.97%), FPR (0.18-0.090%), Testing Time (0.53-061 ms/message)	NA

Table A.4: Summary of Others/Hybrid approaches reviewed in [4].

List of Acronyms

ECU	Electronic Control Unit
CAN	Controller Area Network
OBD	On-Board Diagnostics
DBC	Communication Database for CAN
SoF	Start of Frame
CAN-ID	CAN Identifier
RTR	Remote Transmission Request
IDE	Identifier Extension Bit
DLC	Data Length Code
CRC	Cyclic Redundancy Check
ACK	ACKnowledge
EoF	End of Frame
IFS	Inter-Frame Space
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
T&FA	Time and Frequency Analysis
CI&IT	Computational Intelligence and Information Theory
DT-DS	Decision Tree Ensemble-based Intrusion Detection System
BFF-IDS	Blockchain-based Federated Forest IDS
AI	Artificial Intelligence
ML	Machine Learning

LOF	Local Outlier Factor
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
DNN	Deep Neural Network
LSTM	Long Short-Term Memory
GPT	Generative Pretrained Transformer
NLL	Negative Log-Likelihood
ELU	Exponential Linear Unit
FL	Federated Learning
GDM	Gradient Descent with Momentum
GDM/AG	Gradient Descent with Momentum and Adaptive Gain
DT	Decision Tree
RT	Random Tree
RF	Random Forest
IF	Isolation Forest
SGD	Stochastic Gradient Descent
NB	Naive Bayes
GAN	Generative Adversarial Network
MLP	MultiLayer Perceptron
SOM	Self-Organizing Map
MLHC	Multi-Labeled Hierarchical Classification
TLMD	Two Layers Multi-class Detection
SLMC	Single Layer Multi-class Classification
GRU	Gated Recurrent Unit
AGRU	Attention-based Gated Recurrent neural network

GMM	Gaussian Mixture Model
SVM	Support Vector Machine
OCSVM	One Class Support Vector Machine
K-NN	K-Nearest Neighbors
YOLO	You Only Look Once
VAR	Vector AutoRegression
MBA	Modified Bat Algorithm
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
DR	Detection Rate
TPR	True Positive Rate
TNR	True Negative Rate
FPR	False Positive Rate
FNR	False Negative Rate
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
MCC	Matthews Correlation Coefficient
FPBA	False Positive Before Attack
TTD	Time To Detection
TTP	Testing Time per Packet
FPS	Frames Per Second
FLOPS	FLoating point Operations Per Second
PCC	Pearson Correlation Coefficient
IG	Information Gain

FCBF	Fast Correlation-Based Filter
KPCA	Kernel Principal Component Analysis
IFS	Improved Feature Selection
IDEC	Improved Deep Embedded Clustering
HDL	Hardware Description Language
ASIC	Application-Specific Integrated Circuit
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
CPU	Central Processing Unit
CLB	Configurable Logic Block
LUT	Look-Up Table
MPSoC	MultiProcessor System on a Chip
DSP	Digital Signal Processor
PE	Processing Engine
DPU	Deep learning Processing Unit
DoS	Denial of Service
MI	Message Injection
CWT	Continuous Wavelet Transform
WCS	Worst Case Scenario
API	Application Programming Interface
ORNL	Oak Ridge National Lab

List of Figures

1.1 CAN structure . . . 1

2.1 Original and extended CAN frame format . . . 7
2.2 Reactive and proactive countermeasure lifecycles . . . 9

3.1 Categories of CAN IDSs. . . . 13
3.2 Classification methods comparison: (a) SLMC model; (b) TLMD model; (c) MLHC model [63]. . . . 21
3.3 NeuroCAN network structure [7]. . . . 23
3.4 Distribution of feature sets. . . . 24
3.5 Comparison of dataset distribution with [4]. . . . 24
3.6 Distribution of attack types considered in the reviewed papers and comparison with results of [4]. . . . 27
3.7 Distribution of reviewed papers using benchmark models. . . . 30

4.1 Comparison of detection techniques distribution with [4]. . . . 31
4.2 IDS placements . . . 35

5.1 Basic CLB structure. . . . 40
5.2 *Vitis AI* overview from [1]. . . . 43
5.3 Structure comparison of: (a) Reduced *CANnolo* [59] and (b) Tsantikidou et al. [78]'s model. . . . 46

List of Tables

4.1 Summary of Flow-based approaches. 32
4.2 Summary of Payload-based approaches. 33
4.3 Summary of Hybrid approaches. 34

A.1 Summary of Flow-based approaches reviewed in [4]. 62
A.2 Summary of Payload-based approaches reviewed in [4]. Part 1 63
A.3 Summary of Payload-based approaches reviewed in [4]. Part 2 64
A.4 Summary of Others/Hybrid approaches reviewed in [4]. 64