Intrusion detection with deep learning for in-vehicle networks

Natasha Al- Khatib

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAT009

Thèse de doctorat

Intrusion Detection with Deep Learning for In-Vehicle Networks

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626 de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Réseaux, Informations et Communications

Thèse présentée et soutenue à Palaiseau, le 24 Mars 2023, par

NATASHA ALKHATIB

Composition du Jury :

Guy Gogniat Professeur, Université Bretagne Sud (LAB-STICC), France	Rapporteur, Président
Guillaume Hiet Professeur, Centrale Supelec, France	Rapporteur
Rim Moalla Architecte système, Renault, France	Examinatrice
Ons Jelassi Maître de conférences, Télécom Paris, France	Examinatrice
Marc-Oliver Pahl Professeur, IMT Atlantique, France	Examinateur
Jean-Luc Danger Professeur, Télécom Paris (LTCI), France	Directeur de thèse
Hadi Ghauch Maître de conférences, Télécom Paris (LTCI), France	Co-encadrant de thèse

Abstract

In-vehicle communication which refers to the communication and exchange of data between embedded automotive devices plays a crucial role in the development of intelligent transportation systems (ITS), which aim to improve the efficiency, safety, and sustainability of transportation systems. The proliferation of embedded sensor-centric communication and computing devices connected to the in-vehicle network (IVN) has enabled the development of safety and convenience features including vehicle monitoring, physical wiring reduction, and improved driving experience. However, with the increasing complexity and connectivity of modern vehicles, the expanding threat landscape of the IVN is raising concerns. A range of potential security risks can compromise the safety and functionality of a vehicle putting the life of drivers and passengers in danger.

Numerous approaches have thus been proposed and implemented to alleviate this issue including firewalls, encryption, and secure authentication and access controls. As traditional mechanisms fail to fully counterattack intrusion attempts, the need for a complementary defensive countermeasure is necessary. Intrusion Detection Systems (IDS) have been thus considered as a fundamental component of every network security infrastructure, including IVN. Intrusion detection can be particularly useful in detecting threats that may not be caught by other security measures, such as zero-day vulnerabilities or insider attacks. It can also provide an early warning of a potential attack, allowing car manufacturers to take preventive measures before significant damage occurs.

The main objective of this thesis is to investigate the capability of deep learning techniques in detecting in-vehicle intrusions. Deep learning algorithms have the ability to process large amounts of data and recognize complex patterns that may be difficult for humans to discern, making them well-suited for detecting intrusions in IVN. However, since the E/E architecture of a vehicle is constantly evolving as new technologies and requirements emerge, we propose different deep learning-based methods for different E/E architectures and for various tasks including anomaly detection and classification. Overall, the thesis' contributions fall into two topics.

Firstly, we propose an application of deep learning-based language models for natural language processing tasks mainly in-vehicle intrusion detection. Inspired by the outstanding performance of bidirectional encoder representations from transformers (BERT) for improving many natural language processing tasks, we develop “CAN-BERT", a deep learning-based network intrusion detection system, to detect cyber-attacks on CAN bus protocol. We show that the BERT model is able to learn the sequence of identifiers (IDs), on the CAN bus for anomaly detection, using the “masked language model" - an unsupervised learning method. Drawing on the remarkable detection results, we ask whether a BERT-based multi-agent IDS can detect attacks that affect the inter-dependencies between asynchronous signals carried by distinct CAN IDs.

Secondly, we propose IDSs for anomaly detection and attack classification for the Ethernet-based protocol - a recent network protocol for vehicles, gaining increasing momentum in standards for connected vehicles. We have thus considered two safety-critical application layer protocols namely the **Scalable-service Oriented Middleware Protocol (SOME/IP)** and the **Audio-Video Transport Protocol (AVTP)**. The first contribution consists in generating a labeled dataset for the SOME/IP protocol - as no standard public dataset is available. The generated dataset includes different types of attacks, and a range of normal activity to provide a baseline for comparison. Once available, we leverage the dataset to devise offline and real-time IDS using a supervised learning approach that is particularly useful for attack classification, as it allows the trained model to learn the characteristics of different types of attacks and to accurately predict the type of attack based on these characteristics. Our second contribution lies in detecting attacks on the AVTP protocol used to transmit audio, video, and other multimedia data over Ethernet networks. We thus compare several unsupervised learning algorithms that can potentially be used to identify unknown anomalous or suspicious behavior in AVTP traffic logs.

In summary, our thesis provides an evaluation of different deep learning techniques for different intrusion detection tasks, i.e., classification and detection. In addition, we show their effectiveness in improving the security of in-vehicle networks including Automotive Ethernet and CAN. With respect to the IVN environment, the proposed IDSs are effective at detecting and responding to potential security automotive threats, due to their low false positive and miss detection rates, and their ability to operate both in offline and real-time constraints.

Résumé

La communication embarquée, qui fait référence à la communication et à l'échange de données entre les dispositifs automobiles embarqués, joue un rôle crucial dans le développement des systèmes de transport intelligents (ITS), qui visent à améliorer l'efficacité, la sécurité et la durabilité des systèmes de transport. La prolifération des dispositifs informatiques et de communication embarqués centrés sur les capteurs et connectés au réseau embarqué (IVN) a permis le développement de fonctions de sécurité et de commodité, notamment la surveillance du véhicule, la réduction du câblage physique et l'amélioration de l'expérience de conduite. Cependant, avec la complexité et la connectivité croissantes des véhicules modernes, l'expansion du paysage des menaces du RVI suscite des inquiétudes. Une série de risques de sécurité potentiels peuvent compromettre la sécurité et la fonctionnalité d'un véhicule, mettant en danger la vie des conducteurs et des passagers.

De nombreuses approches ont donc été proposées et mises en œuvre pour pallier ce problème, notamment les pare-feu, le cryptage, l'authentification sécurisée et les contrôles d'accès. Comme les mécanismes traditionnels ne parviennent pas à contrer complètement les tentatives d'intrusion, il est nécessaire de mettre en place une contre-mesure défensive complémentaire. Les systèmes de détection d'intrusion (IDS) ont donc été considérés comme un élément fondamental de toute infrastructure de sécurité réseau, y compris le RVI. La détection d'intrusion peut s'avérer particulièrement utile pour détecter les menaces qui ne sont pas forcément prises en compte par d'autres mesures de sécurité, comme les vulnérabilités de type "zero-day" ou les attaques de l'intérieur. Elle peut également fournir une alerte précoce d'une attaque potentielle, permettant aux constructeurs automobiles de prendre des mesures préventives avant que des dommages importants ne se produisent.

L'objectif principal de cette thèse est d'étudier la capacité des techniques d'apprentissage profond à détecter les intrusions à bord des véhicules. Les algorithmes d'apprentissage profond ont la capacité de traiter de grandes quantités de données et de reconnaître des modèles complexes qui peuvent être difficiles à discerner pour les humains, ce qui les rend bien adaptés à la détection des intrusions dans les IVN. Cependant, étant donné que l'architecture E/E d'un véhicule évolue constamment avec l'apparition de nouvelles technologies et exigences, nous proposons différentes méthodes basées sur l'apprentissage profond pour différentes architectures E/E et pour diverses tâches, notamment la détection d'anomalies et la classification. Globalement, les contributions de la thèse se répartissent en deux thèmes.

Premièrement, nous proposons une application des modèles de langage basés sur l'apprentissage profond pour des tâches de traitement du langage naturel, principalement la détection d'intrusion dans les véhicules. Inspirés par les performances exceptionnelles des représentations d'encodeurs bidirectionnels à partir de transformateurs (BERT) pour améliorer de nombreuses tâches de traitement du langage naturel, nous développons "CAN-BERT", un système de détection d'intrusion réseau basé sur l'apprentissage profond, pour détecter les cyber-attaques sur le protocole du bus CAN. Nous montrons que le modèle BERT est capable d'apprendre la séquence d'identifiants (IDs), sur le bus CAN pour la détection d'anomalies, en utilisant le "modèle de langage masqué" - une méthode d'apprentissage non supervisée. En s'appuyant sur les résultats remarquables de détection, nous nous demandons si un IDS multi-agent basé sur BERT peut détecter des attaques qui affectent les interdépendances entre les signaux asynchrones transportés par des ID CAN distincts.

Deuxièmement, nous proposons des IDS pour la détection des anomalies et la classification des attaques pour le protocole basé sur Ethernet - un protocole réseau récent pour les véhicules, qui prend de plus en plus d'ampleur dans les normes pour les véhicules connectés. Nous avons ainsi considéré deux protocoles de couche d'application critiques pour la sécurité : le **Scalable-service Oriented Middleware Protocol (SOME/IP)** et le **Audio-Video Transport Protocol (AVTP)**. La première contribution consiste à générer un jeu de données étiquetées pour le protocole SOME/IP, car aucun jeu de données public standard n'est disponible. Le jeu de données généré comprend différents types d'attaques et une gamme d'activités normales afin de fournir une base de comparaison. Une fois disponible, nous exploitons le jeu de données pour concevoir des IDS hors ligne et en temps réel à l'aide d'une approche d'apprentissage supervisé particulièrement utile pour la classification des attaques, car elle permet au modèle formé d'apprendre les caractéristiques des différents types d'attaques et de prédire avec précision le type d'attaque en fonction de ces caractéristiques. Notre deuxième contribution réside dans la détection des attaques sur le protocole AVTP utilisé pour transmettre des données audio, vidéo et autres données multimédia sur les réseaux Ethernet. Nous comparons ainsi plusieurs algorithmes d'apprentissage non supervisés qui peuvent potentiellement être utilisés pour identifier des comportements anormaux ou suspects inconnus dans les journaux de trafic AVTP.

En résumé, notre thèse fournit une évaluation de différentes techniques d'apprentissage profond pour différentes tâches de détection d'intrusion, c'est-à-dire la classification et la détection. En outre, nous montrons leur efficacité dans l'amélioration de la sécurité des réseaux embarqués, notamment Automotive Ethernet et CAN. En ce qui concerne l'environnement IVN,

les IDS proposés sont efficaces pour détecter et répondre aux menaces potentielles de sécurité automobile, en raison de leur faible taux de faux positifs et de détection manquée, et de leur capacité.

"The inability to predict outliers implies the inability to predict the course of history"
— Nassim Nicholas Taleb, The Black Swan: The Impact of the Highly Improbable

Acknowledgments

This thesis would not have been possible without the help and input of many people.

First of all, I would like to thank my supervisor Professor Jean-Luc Danger, and my co-supervisors Doctor Maria Mushtaq and Doctor Hadi Ghauch for providing support and sharing their expertise. Thanks to you, I had very helpful discussions regarding the development of the intrusion detection model for automotive security. You have helped me become a better researcher and communicator, and I could not have done this without your patience, expertise, guidance, and feedback throughout the research and writing processes.

I would like to express my deepest gratitude for the generous support of the Chaire Connected Cars Cybersecurity (C3S) and their funding towards my doctoral thesis. Your belief in the significance and potential impact of my research has been instrumental in bringing this project to fruition. I am grateful for the opportunities your funding has provided.

I would like to thank the jury members Professor Guy Cogniat, Professor Guillaume Hiet, Professor Marc-Oliver Pahl, Doctor Rim Moalla, and Doctor Ons Jelassi for the countless hours you have dedicated to reviewing my work and providing valuable feedback. Your insightful comments and probing questions have played a vitole role in shaping my understanding and refining my arguments.

I would like to thank my best friend Lina Achaji for sparking my initial passion for Cybersecurity and empowering me throughout my undergraduate courses to pursue that passion. You have been the best and most encouraging friend and colleague I could ever ask for; I wouldn't be here without you. Additionally, I would like to give a shoutout to my colleagues and friends Sana, Aya, Iman, Chadi, Mahmoud, Lara, Mustapha, Jihane, Amar, Zulal, Lilia, Clement, Paolo, Mehrasa, Akram, Abdelaziz, Muath, Majd, Mohamad, Taghrid, Maryam, Luis, Akram, Thomas as well as many others for their company and friendships. We had a lot of fun times together.

Finally and foremost, I would like to thank my amazing parents Raed and Rana for giving me all the support and encouragement I could have hoped for. To my siblings Amro and Céline, thank you for embodying what I find good in this world. I also owe a special thanks to my family members Fatmeh, Hisham, Bushra, Wassim, Jinane, Hadi, Bashir, Nour, Chadi for your continued support throughout my time at Télécom Paris. To Ghida, I cannot express the impact

you had on my time at Télécom Paris and my entire life. Your unconditional love has led me to where I am today, and I realize how lucky of a person this makes me.

Natasha Alkhatib
Massy, March 2023

Contents

List of Figures

List of Tables

LIST OF TABLES

List of Abbreviations

ADAS Advanced Driving Assistance System.
AUC Area Under Curve.
AVB Audio Video Bridging.
AVTP Audio Video Transport Protocol.

BERT Bidirectional Encoder Representations From Transformers.

CAN Controller Area Network.
CNN Convolutional Neural Networks.

DoIP Diagnostic communication over Internet Protocol.

ECU Electronic Control Unit.

FFN Feedforward Neural Networks.

HIDS Host-based IDS.

IDS Intrusion Detection System.
IF Isolation Forest.
IVN in-vehicle network.

LIN Local Interconnect Network.
LOF Local Outlier Factor.
LSTM Long short-term memory neural networks.

MOST Media Oriented System Transport.

NIDS Network-based IDS.
NLP Natural Language Processing.

OCSVM	One-class SVM.
RNN	Recurrent Neural Networks.
ROC	Receiver Operating Characteristic.
RPC	Remote Procedure Call.
SOC	Security Operations Center.
SOME/IP	Scalable service-Oriented MiddlewarE Protocol.
UDP-NM	User Data Protocol Network Management.
V2I	Vehicle-to-Infrastructure.
V2V	Vehicle-to-Vehicle.
V2X	Vehicle-to-Everything.

Part I

Introduction and Background

CHAPTER 1

Introduction and Motivation

This chapter provides an introduction to the thesis by first discussing the background and context of the research work. We highlight the worst-case in-vehicle intrusions caused by malicious attackers manipulating diverse automotive functionalities. Striving to build defensive techniques generally applicable to a variety of in-vehicle networks, we present our research aim and corresponding research contributions.

Contents

1.1 Context: Can we trust modern vehicles?

The automotive industry is undergoing rapid technological change as vehicles are becoming intelligent computers on wheels. In pursuit of autonomy, modern vehicles witnessed a proliferation

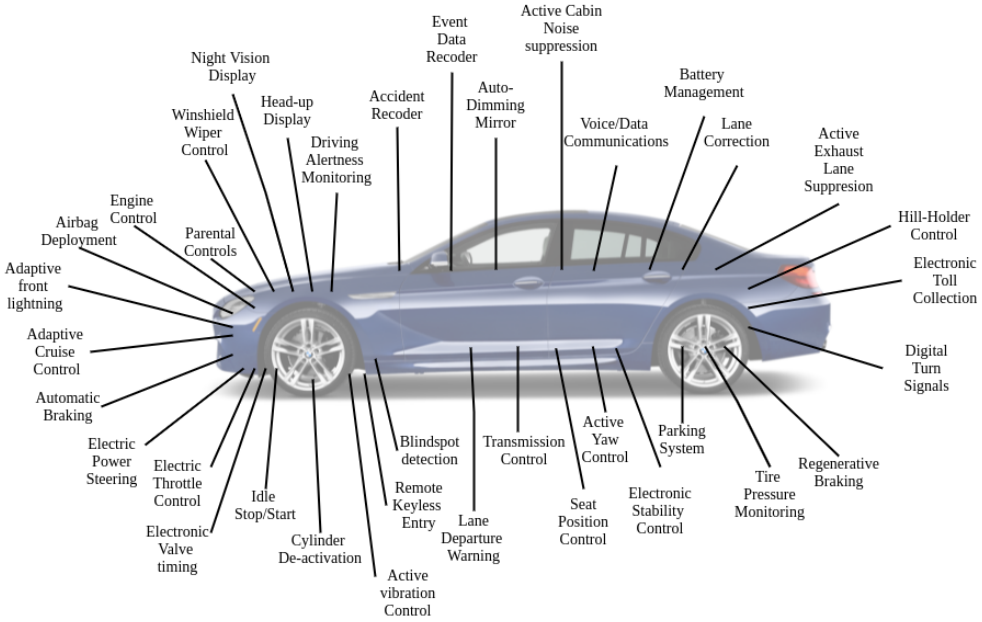

Figure 1.1: Available ECUs in modern vehicles.

in the number of embedded controllers known as Electronic Control Unit (ECU) networked throughout the body of a car (presented in Figure 1.1), in the complexity of their accompanying software, and in the number of short-range (e.g. Wi-Fi, Bluetooth, NFC, etc.) and long-range (e.g. Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Everything (V2X), etc.) communication interfaces to the external world. Moreover, the integration of IVN has led to intricate electronic orchestration between various components (e.g. sensors, actuators, ECU) as well as software choreography, thereby arising advanced functionalities. Ten years ago, only premium cars contained 100 microprocessor-based ECU networked throughout the body of a car, executing 100 million lines of code or more. Today, high-end cars like the BMW 7-series with advanced technology like advanced driver-assist systems (ADAS) may contain 150 ECU or more, while pick-up trucks like Ford's F-150 top 150 million lines of code. Even low-end vehicles are quickly approaching 100 ECU and 100 million lines of code as more features that were once considered luxury options, such as adaptive cruise control and automatic emergency braking, are becoming standard [1].

1.1.1 Background on IVNs

Historically, vehicle manufacturers have leveraged dedicated wires between sensors and actuators to deliver automotive functions. However, as more and more electronics are taking over the driver functions, the vehicle would choke in wiring and several problems will arise in production, in finding space, and for both reliability and troubleshooting, [100]. For an average well-tuned vehicle, every extra 50 kilograms of wiring—or extra 100 watts of power—increases fuel consumption by 0.2 liters for every 100 kilometers traveled [89]. Several IVN (e.g. Controller Area Network (CAN), FlexRay, MOST, Automotive Ethernet) with different communication mechanisms and protocols have thus been proposed to counter the problems of point-to-point wiring mass. Their main role lies in interconnecting various components (e.g., including ECU, gateways, sensors, and actuators) leading to weight, space, and cost reduction. By being a fundamental technical resource for innovativeness, they allow the real-time exchange of data (e.g., sensor measurements, diagnostic messages, etc) and resources among the distributed application. As depicted in Figure. 1.2, a variety of communication technologies prevail in

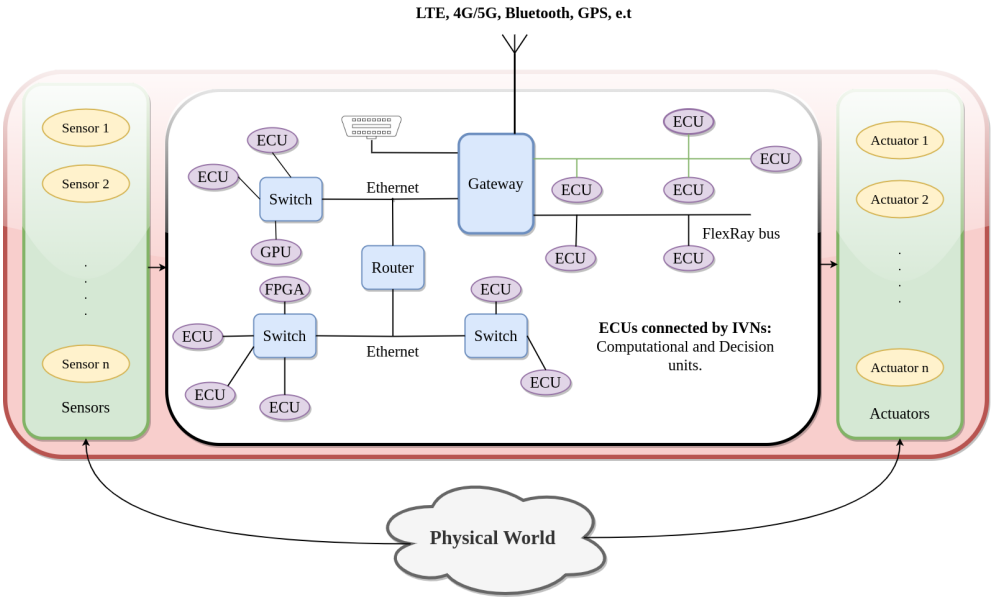

Figure 1.2: Overview of the overall IVN technologies, Source [135].

modern vehicles and exchange information through a central gateway. As presented in Table.1.1, these technologies differ in terms of supported data rate, adopted communication mechanism, and robustness methods. A survey performed in 2015 showed that major car manufacturers

Technology	Multiple Access Scheme	Data rate	Robustness	Target use case
CAN(FD)	Priority-based messages	Generally 500 kbps (2 Mbps) shared	Differential signal, comparably small data rate	Robust ECU control
LIN	Master-Slave and schedule tables	19.2 kbps shared	Small data rate	Low-cost control
MOST	Priority-based, TDMA, token	$\leq$ 25,50,150 Mbps shared	Optical for MOST25/150	Complex, high-end audio
FlexRay	(Flexible) TDMA	$\ll$ 10 Mbps shard	Differential signal	Real-time control, X-by-Wire
Automotive Ethernet	Switched, for each link on the network, queuing	100/1000 Mbs per link and direction	Differential signal, intelligent modulation, and filtering	High data rates, case-independent packets

Table 1.1: Comparison of existing in-vehicle technologies, Source [100]

use on average 8 different digital communication systems inside their cars today [100]. A more detailed discussion regarding the role of each technology is presented in Chapter 2. The central gateway and/or other ECUs interact seamlessly with the external world (e.g., other vehicles, infrastructure, and pedestrians) through communication interfaces. A network-enabled vehicle has thus become a node in the Intelligent Transportation System which contributes to improving the safety, efficiency, and mobility of the transportation system. However, with the increasing connectivity, exploiting the vulnerabilities of the IVN and conducting cyberattacks against vehicles has become an understandable concern. If an attacker succeeds in tampering with the driving functions (e.g., acceleration, braking, steering), the in-vehicle network is impacted and traffic safety is thus compromised. We present in the following section major attacks against vehicles.

1.1.2 Automotive Attacks

As depicted in Figure. 1.3, there are numerous examples demonstrating the success of hackers in conducting attacks against modern vehicles. At the Black Hat security conference, automotive cybersecurity researchers presented novel attacks that target the Jeep Cherokee through Wi-Fi connection [103]. Additionally, they have achieved the invasion of the Toyota Prius and Ford Escape. In 2017 and again in 2018, whitehat hackers from the Keen Security Lab achieved the remote control on Tesla Model S in both Parking and Driving Mode [107]. The hack involved remote attacks which compromised CAN Bus to achieve remote control on Tesla cars. In 2019, Cai et al. [35] revealed the vulnerabilities that existed in three vehicle components of BMW cars: NBT Head Unit, Telematic Communication Box, and Central Gateway. These vulnerabilities could have been exploited by an attacker via the vehicle's external-facing I/O

Figure 1.3: Timeline of major threats discovered and attacks launched between 2015 and 2020, Source [121].

interfaces, including USB, OBD-II, and cellular network. In particular, with the Telematic Communication Box that could have been compromised without any user interaction, an attacker could have triggered or controlled vehicular functions remotely over long distances by combining multiple vulnerabilities and thereby sending unsolicited UDS messages via the BMW vehicle's internal CAN bus, whenever the car would have been parked or driven. Furthermore, as vehicles are being progressively equipped with multiple sensors (LiDAR, radar, camera, etc.) enabling local awareness of their surroundings, hackers are also launching attacks against them to lower their data quality [112]. By exploiting a vulnerability caused by software bugs, configuration errors, and weak network design, a hacker can control desired automotive functionalities. Thus, we can classify attacks against modern vehicles into the following categories:

- **Physical attacks:** These attacks aim to modify the normal behavior of the vehicle by targeting its sensors (LiDAR, radar, camera, etc.) including blinding, jamming, and spoofing attacks. In fact, a fully automated vehicle highly depends on its sensors' measures to make short-term (i.e. safety-related) and long-term (i.e. planning) driving decisions [112]. Indeed, any attack that degrades sensor data can cause a false driving reaction. By attacking a camera, it might falsely interpret a speed limit sign, leading to unsafe driving conditions for the vehicle's passengers. If a LiDAR detects a fake obstacle due to an attack and activates an emergency brake, it will seriously modify traffic efficiency if done extensively.

- **Cyberattacks:** Assuming that an attacker has physical or logical access to the in-vehicle network (LIN, CAN, MOST, FlexRay, Ethernet), these attacks aim to circumvent,

completely or partially, their connected safety-critical systems while ignoring completely the driver input. In fact, they are achieved through the transmission of malicious command injections via the in-vehicle network via a broad range of attack surfaces (including CD players, Bluetooth, and cellular radio). In this case, the attacker is either willing to compromise an in-vehicle component such as the telematic controlling unit, or willing to interact with the in-vehicle network.

Despite the proliferation of several attacks that target various automotive devices, we are mainly interested in cyberattacks that depend on IVNs' vulnerabilities to be achieved. Physical attacks are also important and need to be urgently prohibited. However, they are addressed using specific techniques which are out of the thesis' scope.

1.2 Achieving adequate cybersecurity

Due to the intrinsic vulnerabilities of IVN of modern vehicles, a copious amount of attacks are conducted against modern vehicles. Mitigating this emergent risk has thus become a major objective of the UN Economic Commission for Europe (UNECE) which announced in June 2020 the regulations WP.29 R155 [22] and R156 [23] that need to be adopted by the automotive industry, including OEMs and Tier suppliers. These guidelines, combined with the ISO/SAE 21434 standard [118], hold the automotive industry responsible for ensuring the prioritization of cybersecurity throughout the vehicle development, delivery, and entire lifetime while staying away from outlining specific solutions and exact processes. Consequently, a wide array of security controls, (e.g. authentication, encryption, firewalls, and secure updates) have been proposed to guarantee IVN security, mitigate, and prohibit attacks from growing substantially. Through our thesis, we consider developing an IVN-specific Intrusion Detection System (IDS) that is able to monitor suspicious network traffic behavior and detect potential intrusions. However, the circumstances in the IT industry vary from those found in the car. In fact, the implementation of security measures in an IVN is challenging as it has a fixed topology of limited size and resources (memory, computing power,..), is heterogeneous, is mass-produced, and is safety-critical. The characteristics of IDS IVN are thus dependent on the target IVN type and its corresponding constraints. As previously discussed in Section 1.1.1, a vehicle is composed of several IVN types due to automotive function diversification. Hence, to guarantee the overall detection of automotive cyberattacks, the IVN IDS must monitor several network traffic transmitted through various automotive protocols.

1.3 Deep learning for Cybersecurity

Artificial Intelligence (AI) is concerned with building systems that can handle tasks that require intelligence. It covers a large array of methods, including those based on logic, search, and probabilistic reasoning. Machine learning (ML) is an approach to AI that learns from observed data to tackle various tasks by providing statistical estimations of complicated mathematical models. This area has seen spectacular success and is sometimes interchangeably used with AI, however as seen in Figure 1.4 it is just one way to achieve AI systems. A deep neural network is a subset of ML models that have powered several complex tasks and which will be the scope of our work in this thesis.

1.3.1 Deep learning

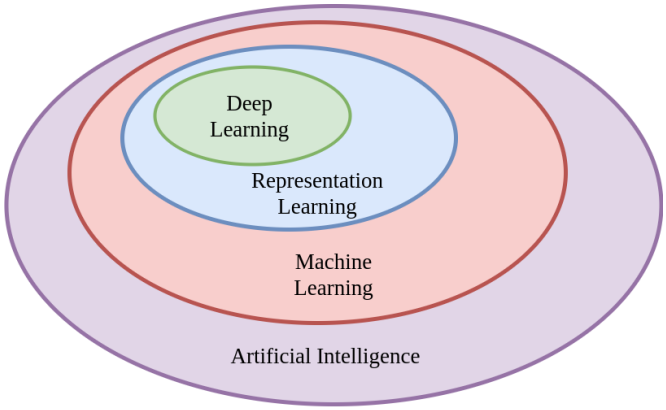

Figure 1.4: A Venn-diagram of artificial intelligence: link between artificial intelligence, machine learning and deep learning.

Deep learning is an approach to Artificial Intelligence (AI) that has become wildly successful over the years due to its ability to reach human performance on numerous complex tasks including computer vision, natural language processing, speech recognition, and others. As illustrated in Figure 1.4, it is also a particular kind of machine learning that enables multi-layered processing models to learn and represent raw sensory input data with hierarchical levels of abstraction emulating the brain's multi-modal information perception and understanding, thus implicitly capturing complex structures of high-dimensional data [51]. It is a large and diverse group of techniques that includes, but is not limited to, neural networks, probabilistic hierarchies, and other algorithms for learning features, both in an unsupervised and supervised setting. The

Milestone/contribution	Contributor,year
MCP model, regarded as the ancestor of the Artificial Neural Network	McCulloch & Pitts, 1943
Hebbian learning rule	Hebb, 1949
First perceptron	Rosenblatt, 1958
Backpropagation	Werbos, 1974
Neocognitron, regarded as the ancestor of the Convolutional Neural Network	Fukushima, 1980
Boltzmann Machine	Ackley, Hinton & Sejnowski, 1985
Restricted Boltzmann Machine (initially known as Harmonium)	Smolensky, 1986
Recurrent Neural Network	Jordan, 1986
Autoencoders	Rumelhart, Hinton & Williams, 1986 Ballard, 1987
LeNet, starting the era of Convolutional Neural Networks	LeCun, 1990
LSTM	Hochreiter & Schmidhuber, 1997
Deep Belief Network, ushering the “age of deep learning"	Hinton, 2006
Deep Boltzmann Machine	Salakhutdinov & Hinton, 2009
AlexNet, starting the age of CNN used for ImageNet classifcation	Krizhevsky, Sutskever, & Hinton, 2012
Transformer	Vaswani, 2017

Table 1.2: Important milestones in the history of neural networks and machine learning, leading up to the era of deep learning, Source [51]

deep learning research field dates back to the 1940s when researchers were highly motivated to build computational models that simulate how the biological brain learns. In 1943, McCulloch and Pitts [101] tried to build the first computational model that mimics the functionality of a biological neuron which in turn led to important contributions to the development of artificial neural networks (ANNs). In 2006, Geoffrey Hinton's deep belief network algorithm efficiently trained using greedy layer-wise pretraining [32] for unsupervised learning techniques brought a series of developments also for supervised learning algorithms presented in Table 1.2, including LeNet [87], Long Short-Term Memory [64], and Transformers [127].

Despite being known in the 1940s, deep learning algorithms have only recently seen tremendous growth in their popularity and usefulness due to several factors that enabled training deeper networks including the abundance of large, high-quality, publicly available labeled datasets and the advent of parallel GPU computing for fast and efficient training. Additionally, other factors have also played a key role in the application of deep learning to a broader and broader set of applications, such as the alleviation of the vanishing gradient problem owing to the disengagement from saturating activation functions (such as a hyperbolic tangent and the logistic function), the development of new regularization techniques (e.g., dropout, batch normalization, and data augmentation), and the advent of supportive software libraries for important research projects or commercial products like Tensorflow [19], Pytorch [13], etc.

1.3.2 Can deep learning address cybersecurity concerns?

Due to the massive amount of cyberattacks, the cybersecurity field is becoming an arms race — attackers are increasingly creating attacks, and defenders are constantly reacting to new attacks while determining how to enhance defenses against new vulnerabilities. With the unprecedented volume of daily network traffic and malware, security companies and research laboratories have thus decided to devise accurate automated systems able to guarantee the security of computer devices, networks, and software applications against network intrusions and malware infections.

Problem	Definition	Solution
Phishing	Digital theft that disguises itself as legitimate or genuine sources to steal uses' private and confidential information.	Detection [44]
Malware	Programs that are designed to have undesirable or harmful effects on a computer system	Detection [142]
Intrusion	A security event, or a combination of multiple security events, that constitutes a security incident in which an intruder gains, or attempts to gain, access to a system or system resource without having the authorization to do so.	Detection [26]
Vulnerability	A weakness, flaw, or error found within a security system that has the potential to be leveraged by a threat agent in order to compromise a secure network.	Fuzzing [146]

Table 1.3: Deep learning utility for cybersecurity

Inspired by their tremendous success in solving complex tasks for large-data volumes, deep learning techniques have also been proposed to solve existing cybersecurity concerns namely intrusion detection, malware detection, phishing/spam detection, and website defacement detection (presented in Table 1.3). The application of deep learning techniques to cybersecurity has become prominent due to their advantages listed as follows:

- **Simplicity:** In contrast to conventional machine learning (ML) techniques, deep learning significantly simplifies feature handcrafting by replacing brittle, complex, engineering-heavy pipelines with straightforward, end-to-end trainable models, therefore, offloading a substantial amount of labor. In the context of intrusion detection, the ever-changing and expanding cyberthreat environment necessitates exceptional human effort for the custom design of attack-specific features. Deep learning techniques, on the other hand, can be trained for learning the features, making them an attractive option for this task.

- **Scalability:** Traditional ML learning algorithms often require storing all data points in memory, which is computationally infeasible under big data scenarios. In addition, traditional ML algorithms lack scalability since they do not considerably improve their performance with a massive volume of data. In contrast, deep learning models can be

trained on datasets of varying sizes, since they can iterate over small batches of data (e.g., using Stochastic Gradient Descent-SGD). In fact, using a large amount of data when training deep learning techniques prevents model over-fitting.

- **Reusability:** Deep learning models, unlike many traditional ML approaches, can be trained on additional data without starting again from scratch. Consequently, they are suited for continual online training, a desirable property for huge production models. Moreover, trained deep learning models are repurposable and, therefore, reusable via transfer learning, allowing reinvesting of previous work into increasingly sophisticated and robust models. This is crucial in the intrusion detection domain because it reduces the computational and memory requirements of intrusion detection systems when performing multi-task learning applications.

1.4 Deep learning for robust in-vehicle systems

As traditional mechanisms fail to fully counterattack intrusion attempts, the need for a complementary defensive countermeasure is necessary. Intrusion Detection Systems (IDS) have been thus considered as a fundamental component of every network security infrastructure [42], including IVN [135]. The goal of embedded intrusion detection software implemented in any in-vehicle network communication system is to intercept network packets exchanged between interconnected ECUs. Once captured, the packets are analyzed in numerous ways and anomalies are detected. Some IDS devices will detect intrusions simply by comparing the packets and their corresponding network activity to pre-determined attack patterns known as signatures, while others will look for anomalous packet activity representing suspicious behavior. grows stronger with every attack and becomes steadily smarter thanks to a constantly expanding attack signature database. The results are then used by the security experts in deciding the suitable countermeasure to be updated and the IDS, considered as the car's immune system, grows stronger with every attack and becomes steadily smarter thanks to a constantly expanding attack signature database.

Since the proposition of the first IDS [42], numerous approaches have been adopted and thus advanced intrusion detection to its current state for various network infrastructures. Regarding IVN, as the number of ECUs inside vehicles is dramatically increasing, the detection of intrusions occurring on large, and high-dimensional exchanged in-vehicle network traffic is becoming more and more difficult. This is where deep learning techniques come into play and converge with

the IDS realm. As previously mentioned, deep learning has brought tremendous advantages to the cybersecurity field and has proven outstanding performance for the intrusion detection task. Therefore, it would be interesting to check the efficiency of these models in detecting in-vehicle intrusions when faced with the corresponding automotive challenges and constraints from the IVN environment:

- When deployed on available ECUs that are mainly powered by a 32-bit embedded processor, the IDS is constrained by computing power, memory size, and communication capability.
- The E/E architecture is always evolving and new automotive protocols are gaining momentum to enhance the driving experience. Therefore, an IDS must be able to detect intrusions for different kinds of automotive network traffics.
- The cost of automotive manufacturing will increase once the IVN IDS design methodology requires hardware modifications to all ECUs. Therefore, IVN IDS design is subject to cost constraints.
- The IDS must fulfill the high-precision requirements of vehicles such as high attack Detection Rate (DR), a low False Alarm Rate (FAR), and also a high ability to detect unknown attacks.

By respecting the mentioned constraints and overcoming the aforementioned challenges, we can put forward deep learning algorithms to perform timely feature extraction and classification tasks together for IVN intrusion detection.

1.5 Challenges

Monitoring and intrusion detection using deep learning techniques for the existing and novel IVN protocols is challenging for various reasons including the following:

- **Lack of datasets** that represent attacks on the recently deployed automotive protocols, particularly Automotive Ethernet. The ethernet-based protocols Scalable service-Oriented MiddlewarE Protocol (SOME/IP) and Audio Video Transport Protocol (AVTP) protocols are specifically designed for applications in the automotive domain. The benchmark datasets including CIC-IDS2017 [9], CSE-CIC-IDS2018 on AWS [6], CIC DoS dataset (2017) [5] are geared toward typical enterprise networks' protocols like TCP and HTTP and, thus, do not cover attacks on the Automotive Ethernet application layer protocols.

The lack of these datasets and applications providing even basic monitoring and detection support for these protocols requires investigating the optimum features to be learned and thereby the design of the corresponding IDS functionality from scratch.

- **Lack of labels** which are required for the training and evaluation of supervised learning and also for the assessment of semi-supervised and unsupervised intrusion detection tasks. Most of the datasets that are available for IVN intrusion detection are not well labeled and some are not labeled at all. Although some developers provide metadata regarding the generated IVN datasets, they do not label it themselves which can potentially lead to uncertainty in the ground-truth.

- **Imbalanced dataset problem** is a common cybersecurity issue that must be addressed particularly if supervised learning techniques,i.e., for attack detection and classification, will be leveraged. As supervised learning techniques can be advantageous for the detection of specific types of attacks, it is important to find solutions if the dataset cannot be balanced.

- **Challenging structure of IVN data** As different IVN protocols have different communication mechanisms, the proposed deep learning techniques must naturally handle their high dimensional data structure. In the case of CAN, different messages are sent at different times, and thus complex interrelations and physical dependencies between signals of multiple *CAN* identifiers must be well learned by deep learning techniques.

1.6 Thesis Contributions and Outline

The contributions of this thesis deepen our understanding of the utility of deep learning technologies for in-vehicle intrusion detection along three main axes: attack detection using anomaly-based and signature-based approaches, attack classification using signature-based approaches, and non-Natural Language Processing (NLP) for intrusion detection. The overarching research questions that we address are:

How effective are deep learning algorithms when detecting attacks and classifying them for different types of IVN protocols, and what kind of features should we consider w.r.t each network type?

In this first part, we introduce the main topic and in Chapter 2 we cover some necessary

background and theoretical fundamentals. This section shortly outlines the individual contributions that tackle our previously mentioned concerns in Section 1.5.

- In Part II, we study the efficiency of language models when detecting different types of intrusions on CAN bus. Inspired by the outstanding performance of Bidirectional Encoder Representations From Transformers (BERT) for improving many natural language processing tasks, we propose in Chapter 3 “CAN-BERT", a deep learning-based network intrusion detection system, to detect attacks that result in the unusual disappearance of CAN identifiers and the appearance of novel ones on the CAN bus. We show that the BERT model can learn the sequence of arbitration identifiers (IDs) in the CAN bus for anomaly detection using the “masked language model" unsupervised training objective. Drawing on the prominent results, we further investigate in Chapter 4 the ability of a BERT-based multi-agent intrusion detection system to detect stealthier attacks that do not change the timing of CAN identifiers, but rather their carried asynchronous signals. The numerical findings, which were obtained using the benchmark dataset “SynCAN" demonstrate that our proposed model can be leveraged for both frequency-based and payload-based intrusion detection.

- In Part III, we tackle the detection of intrusions on the recently adopted in-vehicle network protocol“Automotive Ethernet". One of the main driving forces behind the adoption of this technology is that it allows the development of new automotive protocols for specific layers within the ISO/OSI models while allowing the reuse of protocols for the remaining others. Hence, in Chapter 5, we investigate the capability of deep learning-based models for offline intrusion detection on SOME/IP application layer protocol. SOME/IP is an automotive middleware protocol that operates at the higher layers of the ISO/OSI layer model. Its key advantages lie in the complexity reduction of the Ethernet-based in-vehicle network by providing serialization, Remote Procedure Call (RPC), and service discovery, among other features. However, no security measures, such as authentication or encryption, are defined in the SOME/IP protocol specification which set the ground for an attacker to exploit a legitimate automotive system and initiate attacks from inside the network. To evaluate the performance of deep learning-based intrusion detection, we construct and label a dataset composed of SOME/IP network packets that mimic both normal and malicious behavior - a significant contribution in the SOME/IP intrusion detection field, as there are no datasets available.

- Next, in Chapter 6, we eschew recurrence in favor of adapting Vaswani's transformer model [127], which permits significantly more parallelization and is, therefore, suitable for the real-time intrusion detection for SOME/IP protocol. Although we use the transformer's model "self-attention" mechanism to enable modeling of the interdependencies between different elements of the input network sequence regardless of their distance, our proposed attention-based architecture "SAID" is considerably simpler as it is not built on the typical Encoder-Decoder architecture format for language translation. For this purpose, we evaluate our proposed approach with two simulated and manually annotated SOME/IP datasets, with different attack ratios, built from the SOME/IP generator tool and which are bigger than the dataset generated in Chapter 5. The results of the extensive experiments indicate that our technique efficiently detects the majority of SOME/IP's protocol violations (AUC = 0.8) within 0.3 ms. A comparative study with various deep learning algorithms is performed to show the outstanding performance of our proposed detector in quality while being more parallelizable and requiring significantly less time to detect intrusions.

- In Chapter 7, we present a comparative analysis of different deep and machine learning-based intrusion detection systems for real-time detection of anomalies on the ethernet-based protocol AVTP. Regarding deep learning-based models, we leverage different types of autoencoders which reconstruct a sequence of exchanged AVTP packets over the in-vehicle network. Anomalies in AVTP packet stream, which may lead to critical interruption of media streams, are therefore detected by computing the corresponding reconstruction error. These models are compared with other state-of-the-art anomaly detection models such as One-class SVM (OCSVM),Local Outlier Factor (LOF), and Isolation Forest (IF). The numerical results, conducted on the recently published **"Automotive Ethernet Intrusion Dataset"**, show that deep learning-based models outperform other baselines under different experimental settings.

- Finally, in Chapter 8 we give concluding remarks and we point out open questions for future research.

1.6.1 Publications

Parts of this thesis contain material previously published in the publications referenced in Table 1.4. Each publication is associated with a project page where you can find additional matters such as the original publication, code, data, presentation slides, and posters.

Ch.	Ref.	Paper context	Conference	Project Page
3	[1]	Self-supervised realtime IDS for CAN	AICCSA	`/canbert/`
4	[2]	Multi-agent IDS for CAN	NDSS VehicleSec	`/amica/`
5	[3]	Supervised offline IDS for SOME/IP	IEMCON	`/offline_ids_someip/`
6	[4]	Supervised realtime IDS for SOME/IP	ICUFN	`/realtime_ids_someip/`
7	[5]	Self-supervised realtime IDS for AVTP	IV	`/realtime_ids_avtp/`

Table 1.4: List of own publications on which this thesis was based including the respective chapter. Each publication is associated with a project page: https://github.com/Alkhatibnatasha

We also present the full list of publications that are published or submitted to conferences during the Ph.D.:

1. N. Alkhatib, et al. "CAN-BERT do it? Controller Area Network Intrusion Detection System based on BERT Language Model." 2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA). IEEE, 2022.

2. N. Alkhatib, M. Mushtaq, H. Ghauch and J. -L. Danger, "AMICA:Attention-based Multi-Identifier model for asynchronous intrusion detection on Controller Area networks", VehicleSec NDSS 2023.

3. N. Alkhatib, H. Ghauch and J. -L. Danger, "SOME/IP Intrusion Detection using Deep Learning-based Sequential Models in Automotive Ethernet Networks," 2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2021, pp. 0954-0962, doi: 10.1109/IEMCON53756.2021.9623129.

4. N. Alkhatib, M. Mushtaq, H. Ghauch and J. -L. Danger, "Here comes SAID: A SOME/IP Attention-based mechanism for Intrusion Detection" , The 14th International Conference on Ubiquitous and Future Networks.

5. N. Alkhatib, M. Mushtaq, H. Ghauch and J. -L. Danger, "Unsupervised Network Intrusion Detection System for AVTP in Automotive Ethernet Networks," 2022 IEEE Intelligent Vehicles Symposium (IV), 2022, pp. 1731-1738, doi: 10.1109/IV51971.2022.9827285.

CHAPTER 2

Background: Deep Learning and IDS

This chapter provides the fundamentals that are used as a basis for the explanations in the later chapters of this work. We give a concise summary of the most widely used in-vehicle networks such as CAN and Automotive Ethernet. Furthermore, intrusion detection systems are briefly discussed. This chapter also highlights the necessary deep-learning algorithms that have been used throughout the thesis.

Contents

2.1 In-vehicle networks

This section provides an overview of the different in-vehicle networks relevant to the work at hand as well as the terms that will be used in later chapters.

2.1.1 Controller Area Network (CAN)

Point-to-point connections were the standard for in-vehicle communication until the early 1990s. However, this strategy quickly failed due to concerns regarding cost, weight, complexity, and reliability as increasingly more ECUs were being integrated into the vehicle. To address this problem, shared communication resources, so-called *busses*, were devised for message exchange between numerous end-nodes along with a set of rules outlining the access to the bus, namely the *communication protocol*. Many communication protocols have thus been established for bus-based communication including Local Interconnect Network (LIN), CAN,Media Oriented System Transport (MOST), or FlexRay. In this thesis, we mainly focus on the CAN bus.

2.1.1.1 CAN Technology

CAN is the primary and the most widely adopted in-vehicle networking technology that was developed at Bosch in 1983 and standardized in 1993 for Low-Speed [10] and High-Speed PHY layers [10]. Unlike other in-vehicle networking technologies, it remains in use. As previously

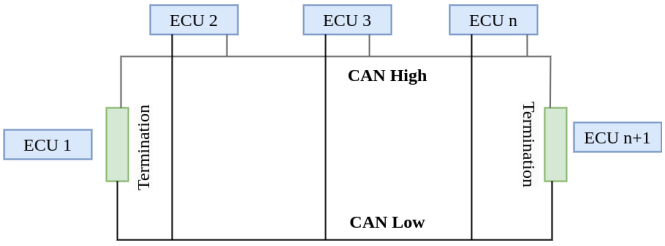

Figure 2.1: CAN-based in-vehicle communication system, Source [100].

mentioned, CAN is a bus system where, as seen in Fig. in 2.1, all attached ECUs share the same wiring for data transmission. This protocol allows numerous ECUs to constantly broadcast data frames consisting of information about the current state of the vehicle. A standard CAN data frame (or packet), featured in Figure 2.2, consists of several fields. The most important fields are described as follows:

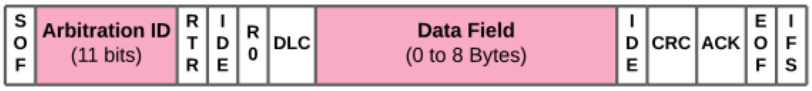

Figure 2.2: CAN frame structure

- **The Arbitration ID:** It is the message header that provides frame identification and is used to perform arbitration, the process by which frames are prioritized when several ECUs are transmitting - the lower the ID, the higher the priority.
- **RTR bit:** If this bit is activated (set to 1), the dataframe must be considered as a remote frame. It is possible for any ECU to request for ID information by transmitting the ID and the RTR bit. In response to this remote frame, the requested identifier and data would be sent back immediately.
- **Data field:** consists of the actual contents of the message, up to 8 bytes, where each piece of distinct information being carried in the message is called a signal. CAN frames with the same ID encode the same set of signals in the same format and are typically sent at a fixed frequency to relay updated signal values. Generally, each ECU is assigned a set of identifiers that only it can transmit. For instance, the PCM can send: ID 0x102 which consists of engine speed, vehicle speed, and odometer signals every 0.05s, and ID 0x45D with signals encrypting the gas pedal and brake pedal angle every 0.01s.

Nonetheless, the CAN protocol lacks security mechanisms and thus needs urgent monitoring which will be discussed in Part II. Additionally, it provides relatively low bandwidths and is not well-suited for the development of upcoming ADAS applications, which rely on large-scale data transmission over the in-vehicle network in order to make crucial driving decisions. To address this bandwidth requirement, while adhering to the stringent cost and weight limits imposed by the automotive industry, attention has turned to Ethernet, a communication protocol that was pioneered for different purposes.

2.1.2 Automotive Ethernet

2.1.2.1 History

The development of Ethernet started between 1973 and 1974 at Xerox PARC in Palo Alto, California by Dr.Bob Metcalfe and Dr.David Boggs [20]. It was commercially introduced in 1980 and was first standardized in 1983 as IEEE 802.3 [24]. This family of wired computer network

technologies represented the primary high-speed technology for the connection of personal computers in local, metropolitan, and wide area networks. The discovery of Ethernet and its subsequent widespread adoption has been a technological triumph, particularly when used for the exchange of information such as the World Wide Web. As a matter of fact, practically every personal computer in the world today communicates over Ethernet.

The word "Ethernet" is commonly used as a general term that refers to a wide broad range of networking technologies and is mostly applied to both the communication protocol and the Ethernet-specific hardware components such as Profinet, Avionics Full-Duplex Switched Ethernet (AFDX), Ethernet Audio Video Bridging (AVB), or Ethernet TSN were developed for usage in certain application domains. They offer industry-specific add-ons and differ in the nuances of the protocol, the hardware used, or the arbitration schemes. Nevertheless, certain features are typically found in Ethernet technology, specifically the frame format, the bandwidths delivered, and the ability to define virtually separate subnets.

2.1.2.2 Towards Automotive Ethernet

1. **Generation 1: Diagnostics over IP** If the in-vehicle network is connected to the diagnostic tester via a 500 kbps high-speed CAN (HS CAN) network, a total software update of a well-equipped, high-end vehicle would have required more than 16 hours [100]. The advent of vehicle Ethernet was thus crucial as car manufacturers desired software updates that required no more than 15 minutes to complete. Connecting the on-board network to the diagnostic tester via 100Base-TX Ethernet over CAT-5 was considered appropriate for timely data reading and software updates, as it delivers sufficient data throughput, is readily available in computers and laptops, and is cost-effective [57].

2. **Generation 2: Driver Assistance Systems and Infotainment** Information and entertainment systems, as well as camera systems, were also studied as potential uses for Ethernet alongside the more common programming and diagnostics use cases. Powered by 100Base-T1 Ethernet technology, Ethernet packets were thus sent at 100 MBps over a single pair of unshielded cables (UTSP), paving the way for sophisticated automotive features which require the deployment of multiple cameras fused with sensor data, such as short and long-range radar. An object detection task that necessitates high-speed communication for the transfer of uncompressed data, for example, can be performed efficiently.

3. **Generation 3: Ethernet as network backbone** In parallel to its advantages for certain vehicle applications, Ethernet can also be employed as a network backbone. In other words, various communication systems can be connected using the switched Ethernet system through its existing and proven IP routing concepts, which allow the transport of a message across domain boundaries to its destination.

2.1.2.3 Advantages of Ethernet in Automotive

With the emergence of sophisticated automotive applications and advanced data-driven functionalities, the communication and bandwidth requirements handled by in-vehicle networks are increasing exponentially. Companies are actually implementing advanced in-car digital entertainment platforms to meet their customer's expectations of video streaming, on-demand content, interactive experiences, local video programming, sports, news, gaming, and much more. Despite their efficiencies, traditional in-vehicle networking technologies,i.e., CAN, FlexRay, and MOST, are considered as limiting factors for innovative automotive functions in terms of bandwidth, packet size, costs, weight, and higher layer protocols. Thus, a migration to a cost-efficient high-speed switched network in modern vehicles that lifts these restrictions is necessary.

Established by the OPEN Alliance, Automotive Ethernet - or more correctly "Ethernet-based communications" - was since then adopted by different car manufacturers thereby bringing in itself a number of fundamental benefits, such as:

- Enabling the exchange of different kinds of data,i.e., videos, images, graphic data, at high data rate (up to 1 Gbps) between interrelated electronic control units (ECU) and which in turn allows automated and autonomous driving functionalities (advanced driving assistance system (ADAS), adaptive cruise control), infotainment services as well as speedy diagnosis and Flash updates.

- Allowing the development of new automotive protocols for specific layers within the ISO/OSI model while enabling the reuse of protocols for the remaining others.

- Changing the in-vehicle networking topologies from decentralized domain-specific architectures to hierarchical with backbone.

- Providing scalability and flexibility for next-generation in-vehicle networking architectures.

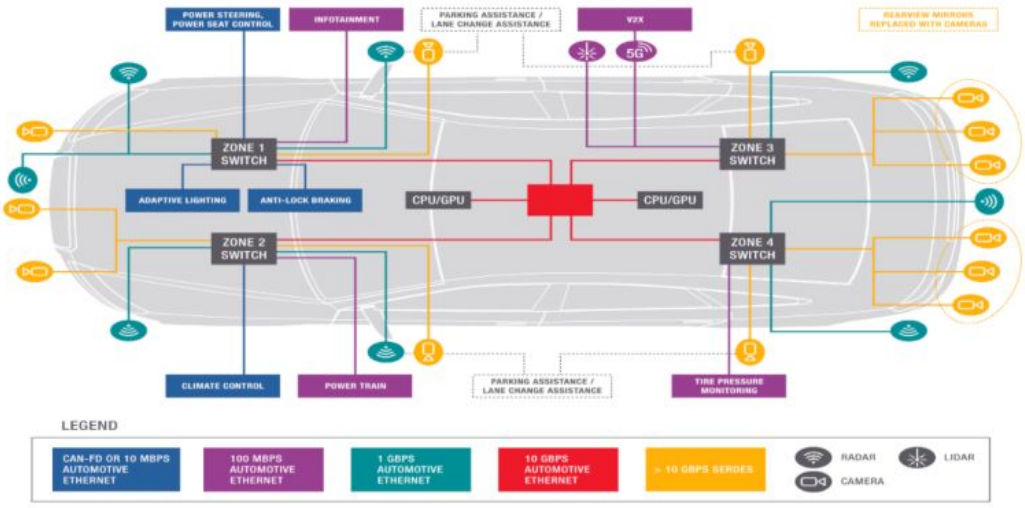

Figure 2.3: Vehicular network

As shown in Fig. 2.3, in vehicular applications, Ethernet is deployed in switched networks that are composed of end nodes and Ethernet switches. End nodes are typically the electronic control units (ECUs) that function as the source(s) and destination(s) of messages transmitted over the network. The primary function that Ethernet switches serve is to transmit data; they do not generate nor process messages of any kind. Ethernet links are used to establish communication between the end nodes and the switches. Each Ethernet link, in contrast to the conventional bus-based communication that is used, for example in CAN networks, functions as a dedicated point-to-point connection between two nodes. Each link consists of a pair of cables that have been twisted together and are attached to the network interfaces of the connection's endpoints.

2.1.2.4 Frame format

Ethernet provides a packet-based transmission, akin to CAN and FlexRay protocols. When a message is considered too large in size, it is thus split into a number of packets. In this case, the separate packets are sent individually in the network and are then merged by the message receiver. As seen in Fig. 2.4, on the physical layer, each packet is composed of a 7-byte preamble, followed by a 1-byte start-of-frame delimiter. The rest of the packet consists of the Ethernet frame itself, which ranges in size from 64 to 1522 bytes, and the 12 bytes of inter-packet space. The structure of the Ethernet frame, which corresponds to the data link layer, is composed of entries consisting of the frame's source and destination addresses (6 bytes

each), the 802.1Q tag (often known as the virtual local area network (VLAN) tag) depicting the VLAN and message priority information, the actual payload, and the 32-bit checksum featuring the cyclic redundancy check. The payload (which also consists of headers and trailers for higher layer protocols such as, for example, the User Datagram Protocol (UDP)) of an Ethernet frame varies between 46 and 1,500 bytes.

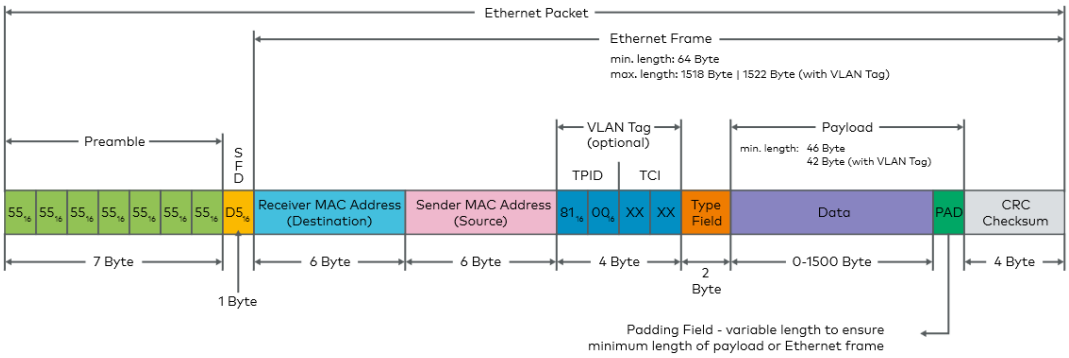

Figure 2.4: Ethernet frame

2.1.2.5 Bandwidth

Ethernet was first conceived using a shared coaxial cable and was used for transmitting 2.94 Megabits of data per second. Ethernet has steadily grown to accommodate more advanced transmission mediums and greater connection lengths as it has risen to become the de-facto network technology. However, the major goal of the Ethernet upgrades was to provide the additional capacity required to deal with the doubling of the world's network bandwidth needs every 18 months. It is unusual to come across anything other than the three most common Ethernet transfer rates of 10 Megabits, 100 Megabits, or 1 Gigabit per second in the realm of consumer electronics. These transfer rates are the prominent ones. However, specialized contexts such as high-performance computing or data centers now function at Ethernet speeds of 10 Gigabits per second, and it is anticipated that they will begin utilizing 100 Gigabits per second in the not-too-distant future.

2.1.2.6 Virtual Local Area Networks

The Ethernet protocol can be used to set up so-called *VLANs* within the network. Each individual VLAN is a specialized subsection of the overall network. In a network with the configured VLANs, the VLAN tag of an Ethernet frame can be used to provide rules for this

frame w.r.t the VLAN areas. In fact, it is possible to make a certain area of the network accessible only to frames within a certain tag, or even to virtually isolate individual areas of the (physically still connected) network from each other by completely preventing frame exchange between them.

2.1.2.7 Network protocols

The deployment of Ethernet-based communication in in-vehicle network systems has several other benefits than the previously listed ones, such as the ability to reuse the associated OSI layers' protocols built and tested in other industries [100] (seen in Fig. 2.5). Furthermore, this cutting-edge technology enables the invention of new protocols for individual layers while reusing protocols for the rest such as the development of the automotive application layer protocol covered in this section.

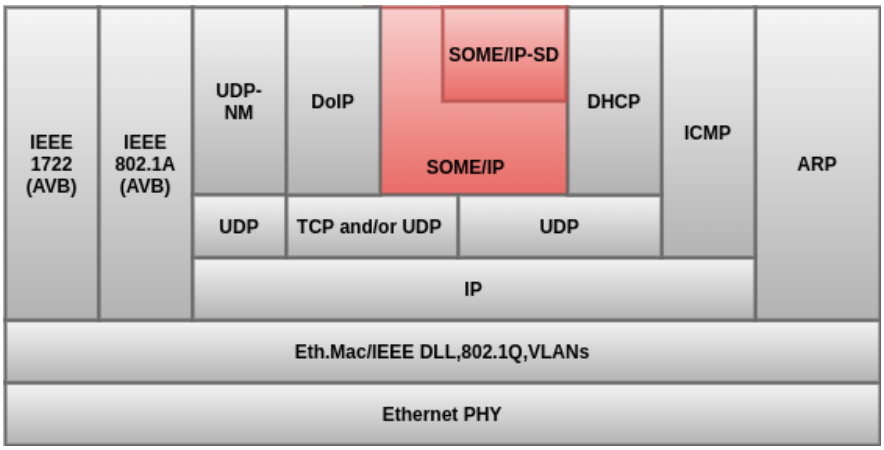

Figure 2.5: Automotive Ethernet protocols, Source [100].

- **Scalable service-Oriented MiddlewarE Protocol (SOME/IP):** The SOME/IP [17], is commonly used for relevant automotive applications due to its service-based communication approach and its adaptability to different automotive operating systems (e.g., QNX, OSEK, and Linux) [100]. In other words, SOME/IP is increasingly adopted to coordinate the exchange of various services between disjoint applications on distinct ECUs.

- **Audio Video Transport Protocol (AVTP):** To ensure low-latency and high-quality transmission of time-critical and prioritized streaming data for high-end infotainment

and ADAS systems, the IEEE 1722 audio-video transport protocol, so-called *AVTP* [4] is adopted. It is used for the transmission of audio, video, and control data transportation on a Time-Sensitive Networking (TSN) capable network [102].

- **Diagnostic communication over Internet Protocol (DoIP):** Remote diagnostics refers to the establishment of automotive diagnostic services that can be carried out remotely, over the air, without the need of connecting a cable directly to a port in the car. As its name implies, the aim of the Diagnostic communication over Internet Protocol (DoIP) protocol is to be able to use existing IP-based networks to carry the diagnostic messages between repair shops and vehicles [7].

- **User Data Protocol Network Management (UDP-NM):** [21] provides the best option to manage the network between ECUs for Ethernet connectivity including performance management, fault analysis, and maintaining the quality of service. It is intended to work together with a TCP/IP Stack, independent of the physical layer of the communication system used.

The introduction of Ethernet into the vehicle also presents new security challenges that are discussed in Part III of the thesis. We specifically focus on the and protocols to provide appropriate security countermeasures. The other protocols listed are equally important but are outside the scope of this work.

2.2 Intrusion Detection System (IDS)

2.2.1 Definition and Role

Firewalls, access control systems, and authentication devices belong to the multi-level security mechanism that must be deployed in modern vehicles in order to successfully secure vehicles against unauthorized access. The detection of intrusions is an additional layer of security, and it has been the primary focus of many recent research investigations. An IDS for IVN is a software program that monitors network traffic packet by packet and analyzes network, transport, and application protocols to detect possible security intrusions. We define a security intrusion as an unauthorized act of bypassing the security mechanisms of a system. If responsive components have been installed in the vehicle's security system, as soon as an intrusion is detected, an alert is triggered and appropriate countermeasures respond to the attack.

2.2.2 Automotive-specific IDS

It is noteworthy that IDSs cannot simply be adopted from the IT environment. On the contrary, automotive-specific requirements, such as restricted computing power, tight memory capacity, and real-time performance, must be taken into consideration [2]. Consequently, a considerable amount of work has been devoted to its standardization in order to facilitate the sharing of information between various development systems. This includes the ISO/SAE 21434 (Road vehicles - Cybersecurity Engineering) [11] which is widely regarded as the future security standard for motor vehicles. Besides other aspects, it requires that vehicle manufacturers must implement a well-defined procedure for handling incidents due to security breaches that have arisen in their vehicles. Yet this can only be accomplished once the vulnerabilities have been identified. This is where the automotive IDS comes into play. The IDS identifies incoming attacks on the vehicle's ECUs and networks, documents them, and then transmits them to a backend at the vehicle manufacturer, known as the . The vehicle manufacturer assesses the data and then determines the appropriate course of action to be taken in response to the attack attempts.

2.2.3 Categories of attacks

We can split IDSs based on their corresponding location and analysis approaches during intrusion detection.

2.2.3.1 Categories of IDSs

The location features where the detection takes place. In this case, we distinguish two types of IDS: network-based IDS, and host-based IDS. A Network-based IDS (NIDS) is placed at a point in the network to monitor packets moving across the network and to check for potential malicious activity. In contrast, a Host-based IDS (HIDS) runs individually at each network component by monitoring the corresponding incoming and outgoing traffic for suspicious activity. One key difference between HIDS and NIDS is the scope of their coverage. HIDS are limited to a single host, while NIDS can monitor traffic across an entire network. Another difference is the type of data each system monitors. HIDS typically monitors system and application logs, as well as system files and resources, while NIDS monitors network traffic, including packets and data streams. Overall, HIDS and NIDS serve complementary roles in a security system. HIDS are effective at protecting individual devices and systems, while NIDS are effective at detecting and responding to threats that span multiple devices or systems. Through our thesis, we mainly devise NIDS for the detection of intrusions on diverse IVN.

2.2.3.2 Detection methods

Depending on the detection method, IDSs typically use one of the following alternative approaches to analyze data during intrusion detection: anomaly-based detection and signature-based detection. Anomaly detection involves monitoring the system or network for unusual or suspicious activity that deviates from previously collected legitimate behavior over a period of time. Typically, these systems use machine learning algorithms to learn what normal behavior entails, then send out notifications to administrators if the system or network represents anomalous activity. Signature-based detection or misuse detection, on the other hand, involves identifying specific data patterns (signatures) that are associated with known threats. In addition, some of these systems compare the current behavior to a set of known attack rules (heuristics) in order to determine whether or not it is malicious. Signature-based detection systems are effective at detecting known intrusions. However, they may not be able to detect novel threats that do not have a matching signature in a previously collected database. Overall, anomaly-based detection is generally considered to be a more proactive approach to detecting threats, as it allows administrators to identify and respond to unusual or suspicious activity before it causes harm. Signature-based detection, on the other hand, is generally considered to be a more reactive approach, as it relies on the existence of known threats and their associated signatures in order to identify and block them. Given this advantage, clearly, anomaly detection would be the preferred approach, were it not for the difficulty in collecting and analyzing the data required, and the high level of false alarms.

2.3 In-vehicle intrusion detection

2.3.1 Problem statement

Given a stream of in-vehicle network packets or data records, the goal of this thesis is to determine network intrusions related to various network protocols, i.e. CAN and Ethernet, efficiently and to overcome corresponding challenges. The temporal relationships between data records are crucial as the sequential ordering of network packets is critical in identifying intrusions. Moreover, each individual record, in this case, is multidimensional and contains the features extracted from the unit of network data (e.g., packet). Thus, this problem can be modeled as a multidimensional stream of records. Both signature-based and anomaly-based approaches have their advantages and disadvantages, which are detailed in 2.2. Using a signature-based method is suitable when constructing an IDS to categorize threats and is even useful when detecting

existing intrusions and repetitive attacks. As new intrusions arise over time, anomaly detection, however, seems to be the best strategy if we want to identify them without categorization. Since the intrusion detection task can be formulated as either an attack detection or classification problem, we investigate the performance of deep learning techniques in addressing both tasks that are considered to be complementary.

With the increasing dimensionality of the considered data instances, many of the traditional attack detection methods become less effective. This is an artifact of the well-known *curse of dimensionality*. In a high-dimensional space, data becomes sparse, and true attack cases are obscured by the noise effects of multiple irrelevant dimensions when analyzed in full dimensionality. To mitigate this issue, we leverage deep learning techniques that are known to be effective for processing high-dimensional datasets.

Additionally, in comparison to the in-vehicle CAN network and other application areas, Ethernet-based embedded network data was not available at the beginning of this thesis [1]. The absence of data, which we regard to be of fundamental relevance for creating up-to-date deep learning models that handle intrusions on modern in-vehicle protocols, was a major challenge that needed to be mitigated as well.

2.3.2 Attack detection

In order to detect attacks or intrusions, we can leverage both signature-based and anomaly-based approaches, as discussed earlier. We consider attack detection as a binary classification (attack/no attack), as opposed to attack classification in which a classifier must classify data instances according to their attack type. From a data-driven perspective, several levels of supervision are possible for the detection of attacks and which we associate with the following two methods:

2.3.2.1 Signature-based approaches

In numerous scenarios, existing examples of in-vehicle network attacks may be available. A subset of the data may be labeled as attacks, while the remainder of the data is regarded as normal. In such cases, the attack detection process is referred to as supervised attack detection, because labels are leveraged for training a model that can determine specific types of attacks. However, it is regarded as a difficult classification problem since labels are extremely unbalanced.

[1] After publishing our SOME/IP dataset, the “Automotive Ethernet" and “TOW-IDS" datasets became accessible for the purpose of intrusion detection on different Automotive Ethernet protocols

Essentially, because attacks are less frequent than normal points, it is possible for standard classifiers to predict that all test points are normal points while achieving excellent accuracy.

2.3.2.2 Anomaly-based approaches

By definition, *anomaly detection* is the process of identifying patterns that do not conform to the expected normal pattern, these patterns are called *anomalies* or *outliers*. For example, in networked systems, we consider the network traffic as anomalous if it shows unusual behavior because of malicious activity. There are various models that have been used to distinguish normal patterns from abnormal data. From a data-driven perspective, we tend to use semi-supervised approaches when only normal data examples are available. However, if the dataset consists of normal data but is contaminated by anomalies that are not known, this problem lends itself to unsupervised techniques.

When leveraging an outlier detection algorithm to a set of data, we can obtain one of the two possible outcomes:

- **Outlier scores:** An outlier score is representative of the “outlierness" level of each data point. Applied to a set of data points, this score can indicate the probability that each of them is an outlier.

- **Binary labels:** A binary label indicates whether or not a data point is an outlier. Although there are techniques for obtaining binary labels directly, it is also possible to transform outlier scores into binary labels. In most cases, this is done by applying thresholds to the outliers; these thresholds are chosen based on the statistical distribution of the scores. While a scoring method provides more information, binary labeling provides only half of it, but the end result is what is often required for decision-making in practical situations.

2.3.3 Attack Classification

In order to classify attacks, we commonly use a signature-based IDS that aims to categorize data instances into their corresponding attack type (multiclass classifier), rather than just two (binary classification). Several binary classifiers have thus been extended to solve multi-class classification problems and are based on neural networks, decision trees, k-nearest neighbors, naive Bayes, and support vector machines. These types of techniques are referred to as adaptation techniques.

2.4 Background: Deep learning

To help flag novel and existing malicious uses of the in-vehicle network, deep learning architectures are leveraged for multiple intrusion detection tasks. These techniques are used to improve detection methods, by creating new rules automatically for signature-based IDS or adapting the detection patterns of anomaly-based IDS. Hence, in this section, we cover the deep learning techniques associated with each task, along with the different learning approaches and the performance metrics used to evaluate the developed systems.

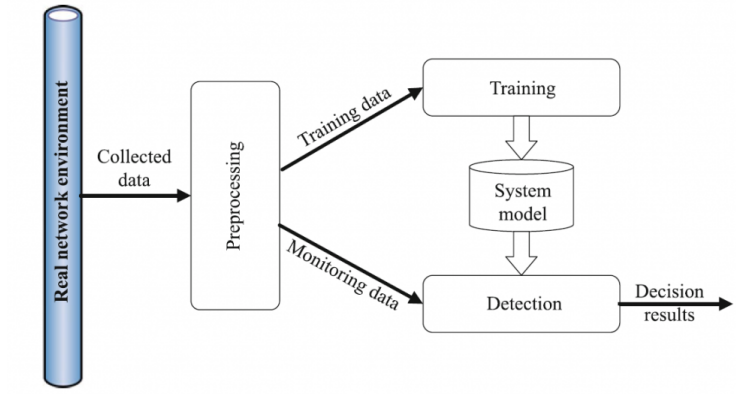

Figure 2.6: Deep learning-based IDS for IVN

To train an anomaly detector, a dataset of normal inputs is sufficient. A classifier, on the other hand, requires a labeled dataset of normal and attack inputs in order to be trained. After training, it can be used to perform intrusion prediction on new incoming network data. As seen in Fig. 2.6, the preprocessed in-vehicle network is fed to the trained IDS, e.g., anomaly detector or attack classifier, which in turn generates a label or an attack score for each input or a sequence of inputs. The label can be binary: normal and attack, or multi-valued indicating various types of attacks. To that end, we distinguish between supervised and unsupervised deep learning methods, next.

2.4.1 Supervised deep learning IDS

Here, we provide a formal description of related supervised deep learning methods, for IDS. The considered supervised learning methods are all characterized by a labeled training set:
$\{ (\mathbf{x}_t , \mathbf{y}_t) \}_{t=1}^{T}$

where the pair ($\mathbf{x}_t$, $\mathbf{y}_t$) is sample $t \in \{1, .., T\}$ of the training set, $\mathbf{x}_t$ is feature vector (of appropriate dimension), $\mathbf{y}_t$ is the binary label vector (of appropriate dimension).

- **Feedforward Neural Networks (FFN):** FFN (a.k.a, multi-layer percepteron, deep neural network) consist of a cascade/composition of the multiple layers of computational units, usually interconnected in a feed-forward way, as shown in Fig. 2.7. Each neuron in one layer has directed connections to the neurons of the subsequent layer.

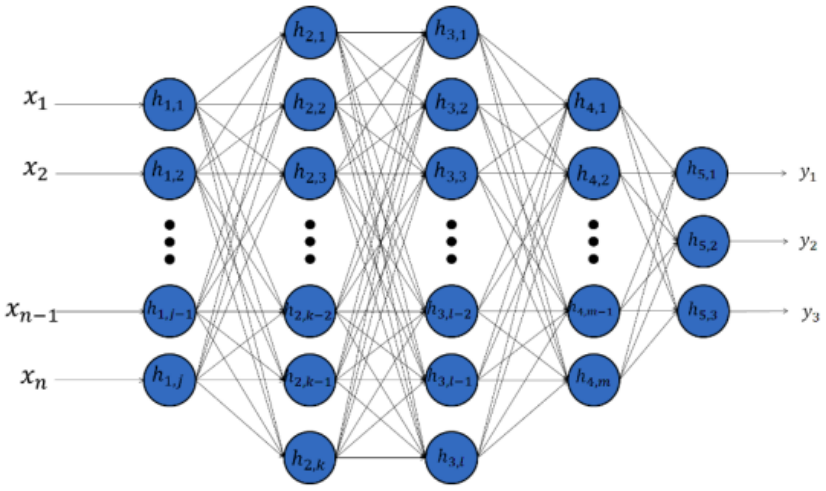

Figure 2.7: Deep neural network.

We consider a genera FFN, as a mathematical compositing of J layers. The equation describing the operation of layer indexed as $j \in \{1, .., J\}$, as follows:

$$\mathbf{z}_j \;=\; \sigma_j(\; \mathbf{W}_j \mathbf{z}_{j-1} \;)\;,\; \forall j \in \{0, ..., J-1\} \tag{2.1}$$

Note that, the bias term b_j is already absorbed in the matrix, $\mathbf{W}_j$. In the above, $\mathbf{W}_j$ denotes the matrix of weights (of a appropriate dimension) for layer $j \in \{1, .., J\}$, $\mathbf{z}_j$ the vector (of a appropriate dimension) representing the input for for layer $j \in \{0, .., J-1\}$, $\mathbf{z}_{j+1}$ the output of layer j that is fed into as input to the next layer $j+1$, and $\sigma_j(\;)$ is the vector-valued vector function (an element-by-element operator) modeling the activation function for layer $j \in \{1, .., J\}$. By recursively substituting the expression in the above equation, we can write the overall input-output equation the FNN, as follows,

$$\mathbf{z}_J \;=\; \sigma_J(\mathbf{W_J} \;\cdots\; \sigma_1(\; \mathbf{W}_1 \mathbf{z}_0 \;)\;) \tag{2.2}$$

Since we are dealing with classification tasks, we choose the activation for the last layer $\sigma_J(-)$, as a *softmax* function. Then, each of the feature vectors $\mathbf{x}_t$, $\forall t \in \{1, ..., T\}$, are fed to the FFN input $\mathbf{z}_0$. Moreover, the FFN output $\widehat{\mathbf{y}}_t$, resulting from sample $\mathbf{x}_t$ is expressed as,

$$\widehat{\mathbf{y}}_t \ = \ \sigma_J(\mathbf{W_J} \ \cdots \ \sigma_1(\ \mathbf{W}_1\mathbf{x}_t\ )\ ) \ , \ \forall t \in \{1, ..., T\}$$

Next, a loss function is defined for sample t, that compares the FFN output ($\widehat{\mathbf{y}}_t$) to the actual value/label ($\mathbf{y}_t$), i.e.,

$$\ell_t(\mathbf{y}_t, \widehat{\mathbf{y}}_t) = \ell_t(\mathbf{y}_t \ , \ \sigma_J(\mathbf{W_J} \ \cdots \ \sigma_1(\ \mathbf{W}_1\mathbf{x}_t\ )\ ))\forall t \in \{1, ..., T\}$$

Note that, $\ell_t(-)$ is a general loss function. However, ℓ_t reduces to the binary cross-entropy (if y_t is a binary scalar), and a multi-class cross entropy loss (if y_t is binary vector). Training the FFN amounts to finding the optimal weight matrices $\mathbf{W}_1^\star, .., \mathbf{W}_J^\star$, which minimize the empirical risk (training error) problem, i.e.,

$$\mathbf{W}_1^\star, .., \mathbf{W}_J^\star = \arg\min_{\mathbf{W}_1,..,\mathbf{W}_J} \ T^{-1} \ \sum_{t=1}^{T} \ \ell_t(\mathbf{y}_t \ , \ \sigma_J(\mathbf{W_J} \ \cdots \ \sigma_1(\ \mathbf{W}_1\mathbf{x}_t\ )\ )) \tag{2.3}$$

Looking at the above problem, while $\mathbf{W}_1, .., \mathbf{W}_J$ are the weights/parameters (to be trained/optimized), all the other quantities are hyper-parameters (not to be optimized). Solving the optimization problem, is done using the backpropagation algorithm. Essentially, a gradient descent algorithm is used to adjust the weights, layer-by-layer. This approach implicitly assumes all the training/testing are sampled in a independent and identically distributed manner.

- **Recurrent Neural Networks (RNN)**: RNN are sequence-based models of key importance for natural language understanding, language generation, video processing, and many other tasks [134].

  In contrast to FFN, a RNN assumes the existence of a hidden temporal or causal relation, among the training/test data. Hence, it belongs to a class of deep learning named sequential models. As seen in Fig. 2.8, the model's input is a sequence of symbols, where at each time step a simple neural network (*RNN unit*) is applied to the current sample, as well as to the network's output from the previous time step. RNN are powerful models, showing

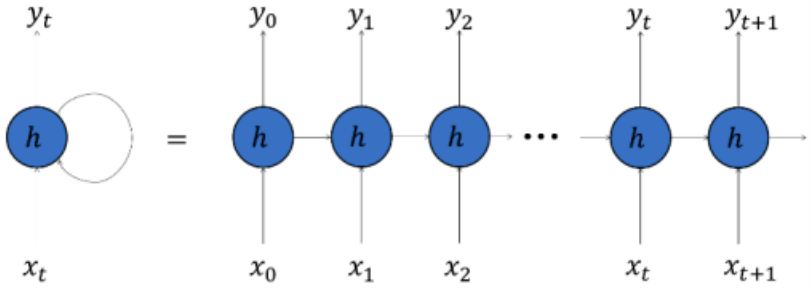

Figure 2.8: Recurrent neural network.

promising performance on many tasks. We will restrict our presentation to a single-layer RNN model. Given the feature vector for sample t $\mathbf{x}_t$ as input to the RNN, a repeated application of function $\mathbf{f}_h$ is performed . With that in mind, the equation describing hidden layer $\mathbf{h}_t$ for sample t,and resulting from feeding $\mathbf{x}_t$ to the RNN input, is

$$\mathbf{h}_t = \mathbf{f}_h(\mathbf{x}_t, \mathbf{h}_{t-1}) = \sigma(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h\mathbf{h}_{t-1} + \mathbf{b}_h) \ , \ \forall t \in \{1, .., T-1\} \tag{2.4}$$

where $\sigma(-)$ an vector-valued vector function applied element-wise. The model output can be defined, for example, as:

$$\widehat{\mathbf{y}}_T = \mathbf{W}_y\mathbf{h}_{T-1} + \mathbf{b}_y \tag{2.5}$$

With that in mind, the rest of the method is similar to the FNN: a loss function is defined (multi-label cross entropy), to from the empirical risk problem. Moreover, for the RNN the weights that are optimized/trained are $\mathbf{W}_h, \mathbf{U}_h, \mathbf{W}_y$. In general, RNNs are more challenging to train than other types of neural networks due to the fact that gradients can easily vanish or explode [**?**]. Training is typically done using the backpropagation through time (BPTT) algorithm. However, recent developments in training and architecture have enabled the development of a variety of RNNs that are simpler to train [64]. In fact, Long Short-Term Memory Units (Long short-term memory neural networks (LSTM)s) were introduced to allow RNNs to handle problems that necessitate long-term memories [65].

- **Long short-term memory neural networks (LSTM)**: Each LSTM unit consists of three gate structures: an input gate, a forget gate, and an output gate. The input and output gates regulate the memory cell's input and output activation, respectively, whilst the forget gate updates the cell's state. The following equations govern the behavior of an LSTM unit:

$$f_t = \sigma(W_{xf} \cdot x_t + W_{hf} \cdot h_{t-1} + b_f) \tag{2.6}$$

$$i_t = \sigma(W_{xi} \cdot x_t + W_{hi} \cdot h_{t-1} + b_i) \tag{2.7}$$

$$\widetilde{C}_t = \text{th}(W_{xa} \cdot x_t + W_{ha} \cdot h_{t-1} + b_a) \tag{2.8}$$

$$o_t = \sigma(W_{xo} \cdot x_t + W_{ho} \cdot h_{t-1} + b_o) \tag{2.9}$$

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \widetilde{C}_t \tag{2.10}$$

$$h_t = o_t \otimes tanh(C_t) \tag{2.11}$$

where h_{t-1} and C_{t-1} are output and cell state at the previous moment, respectively, x_t represents the current input, f represents the forget gate, f_t is a forget control signal which determines if the prior unit's state C_{t-1} should be reserved, $f_t \otimes C_{t-1}$ represents the information retained at the previous moment, i represents the input gate, $\widetilde{C}_t$ is considered as the candidate cell state at time t, i_t represents the control signal for $\widetilde{C}_t$, h_t is regarded as the final output, o_t represents the output control signal. Moreover, $\{W_{x,i}$, $W_{x,f}$, $W_{x,a}$, $W_{x,o}\}$ represents the { input, forget, active, output }-layer connection matrices (all of which to be learned), and $\{W_{h,i}$, $W_{h,f}$, $W_{h,a}$, $W_{h,o}\}$ indicate the { input, forget, active, output }-hidden layer recurrent connection matrices (all to be optimized), σ is the sigmoid activation function and $\otimes$ represents element-wise (Hadamard) product.

- **Convolutional Neural Networks (CNN)** CNNs are popular deep learning techniques used where there is spatial or temporal ordering [86] including a two-dimensional (2D) array of pixels representing a color or grayscale image, three-dimensional (3D) arrays representing videos, and volumetric images.

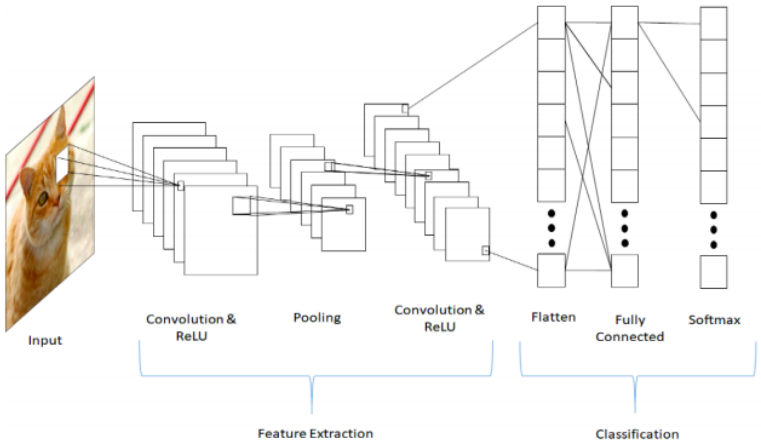

Figure 2.9: Convolutional neural network.

As shown in Fig. 2.9, the architecture of a CNN includes three distinct types of layers: convolution layers, pooling layers, and the classification layer. The convolution layers represent the center core of the CNN. The weights define a convolution kernel applied to the original input, a small window at a time, called the receptive field. The result of applying these filters across the entirety of the input is then passed through a non-linearity, typically a ReLU, and is called a feature map. These convolution kernels, named after the mathematical convolution operation, allow close physical or temporal relationships within the data to be accounted for and help reduce memory by applying the same kernel across the entirety of the image. Pooling layers are used to perform non-linear downsampling by implementing a particular function, such as the maximum, over non-overlapping subsets of the feature map. In addition to reducing the size of the feature maps, and thus the memory needed, pooling layers also reduce the number of parameters, and thus overfitting. Typically, these layers are inserted periodically between convolution layers and then fed into a traditional fully connected DNN. The following equation governs the behavior of both convolutional (also deconvolutional layers):

$$h_k^{[l+1]} = \mathbf{f}(\sum_{j \in J} x_j^{[l]} \circ w_k^{[l]} + b_k), \tag{2.12}$$

where $h_k^{[l+1]}$ is the latent representation of $k-th$ feature map in layer l+1, $\mathbf{f}$ is a non-linear activation function, $x_j^{[l]}$ is the $j-th$ feature map of the output layer in layer l, $w_k^{[l]}$ is the $k-th$ filter weight for the layer l and b_k is the bias parameter, and $\circ$ represents a 2D convolution operation.

2.4.2 Unsupervised and Self-supervised deep learning: anomaly detection

A wide range of classical detection algorithms has been considered, such as the probabilistic method, the distance-based approach, the one-class classification method, etc. Nevertheless, these traditional algorithms often have a tendency to concentrate on lower-dimensional data and struggle when confronted with high-dimensional data, such as images or texts. Additionally, the extraction of features from data in traditional algorithms often involves the use of manual engineering, a process that may be both costly and time-consuming. Recently, deep learning-enabled anomaly detection, so-called *deep anomaly detection*, has emerged as a critical direction that bypasses manual feature engineering and aims at learning feature representations or anomaly scores via neural networks for the sake of anomaly detection [108]. Numerous deep anomaly

detection methods have been introduced, outperforming conventional anomaly detection in addressing challenging detection problems, particularly for the intrusion detection of networked systems. Some of the proposed methods in the thesis fall under the category of unsupervised deep learning. Recall that unsupervised learning consists of unlabeled training/test set,
$\{ \mathbf{x}_t \}_{t=1}^{T}$
where $\mathbf{x}_t$ is feature vector (of appropriate dimension) without having any label.

- **Autoencoders (AE):** This type of approach attempts to learn a low-dimensional feature representation space on which the given instances can be well reconstructed. They are commonly used for data compression or dimension reduction. The heuristic for using this technique in anomaly detection is that the learned feature representations are enforced to learn noteworthy regularities of the data to minimize reconstruction errors; anomalies are difficult to be reconstructed from the compressed space and thus have large reconstruction errors.

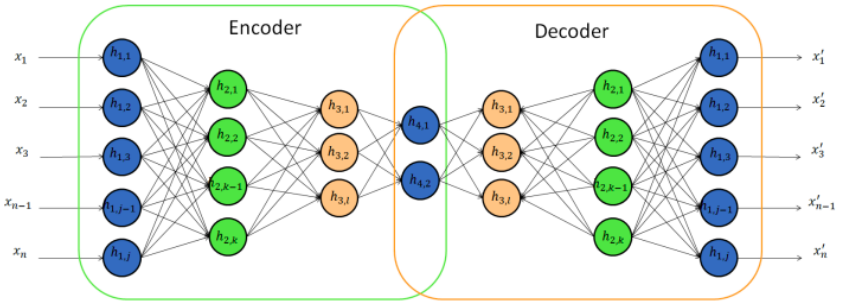

Figure 2.10: Deep autoencoder network.

As seen in Fig. 2.10, an AE is composed of an encoding network and a decoding network. The encoder compresses the original data and projects it to a lower-dimensional feature space, while the decoder attempts to recover the data from the compressed space. The parameters of these two networks are learned with a reconstruction loss function. In order to create low-dimensional representations of the data, a bottleneck network design is often utilized. This architecture requires the model to save the information necessary to construct the data instances. The retained information must be as relevant to the dominant cases, i.e. the normal examples, to reduce the total reconstruction error. Therefore, data examples like anomalies that differ from the norm are not reconstructed as well as they may be. Hence, the inaccuracy in reconstructing the data may be utilized as a direct anomaly score.

The AE may be seen as a composition/cascade of an encoder and decoder. Let ϕ_e be a function modeling the encoding network and ϕ_d is the decoding network. We denote by Θ_e and Θ_d the all the weights parameters/weights of the encoder and decoder, respectively, both of which to be optimized/trained. When sample $\mathbf{x}_t,\ \forall t\{1,...,T\}$ is fed to the AE, the encoder and encoder are governed by the following equations [108]:

$$\mathbf{z}_t = \phi_e(\mathbf{x}_t, \Theta_e) \quad , \quad \widehat{\mathbf{x_t}} = \phi_d(\mathbf{z}_t, \Theta_d), \forall t\{1,...,T\} \tag{2.13}$$

where $\mathbf{z}_t$ for sample $t \in \{1,...,T\}$ is the output of the encoder, that is fed to decoder, and $\widehat{\mathbf{x_t}},\ t \in \{1,...,T\}$ the AE output. Then, the mean-squared error loss function ℓ_t , for sample $t \in \{1,...,T\}$ is defined comparing the true feature vector ($\mathbf{x}_t$) with the corresponding AE output ($\widehat{\mathbf{x}_t}$), as

$$\ell_t = \|\mathbf{x}_t - \widehat{\mathbf{x}}_t\|_2^2 = \|\mathbf{x}_t - \phi_d(\mathbf{z}_t, \Theta_d)\|_2^2 = \|\mathbf{x}_t - \phi_d(\phi_e(\mathbf{x}_t; \Theta_e); \Theta_d))\|_2^2 \ , \ t \in \{1,...,T\}$$

Finally, training is performed by optimizing Θ_e, Θ_d, i.e., minimizing the empirical risk:

$$\{\Theta_e*, \Theta_{d*}\} = \underset{\Theta_e,\Theta_d}{\arg\min}\ (T)^{-1} \sum_{t=1}^{T} \|\mathbf{x}_t - \phi_d(\phi_e(\mathbf{x}_t; \Theta_e); \Theta_d)\|_2^2, \tag{2.14}$$

Given the optimal AE weights, Θ_e*, Θ_{d*} , we compute the anomaly score of each sample $t \in \{1,...,T\}$, as

$$s_{\mathbf{x}_t} = \|\mathbf{x}_t - \phi_d(\phi_e(\mathbf{x}_t; \Theta_e^*); \Theta_d^*)\|_2^2, \forall t \in \{1,...,T\} \tag{2.15}$$

AEs also power the detection of anomalies in data other than tabular data, such as sequence data, graph data, and image/video data by adapting the network to the type of input data, such as CNN-AE, LSTM-AE, Conv-LSTM-AE, and graph convolutional network-AE.

- **Convolutional-based autoencoder (CAE):** Researchers have been widely using CAE for the anomaly detection of concrete defects [39], in automated video surveillance [114], on system logs [41], and on network application protocols such as HTTP [110]. In fact, CAE is composed of convolutional and deconvolutional layers leveraged in the encoder and decoder parts, respectively. In order to use such architecture, input samples must be reshaped into images.

- **Long short-term memory-based autoencoder (LSTM-AE):** Long short-term memory-based Autoencoder (LSTM-AE), widely used for anomaly detection [117][95], is an implementation of autoencoders that uses LSTM as learning layers both in encoder and decoder components, explained earlier.

- **Transformer-based models:** Transformer [127] is a sequence-to-sequence model and

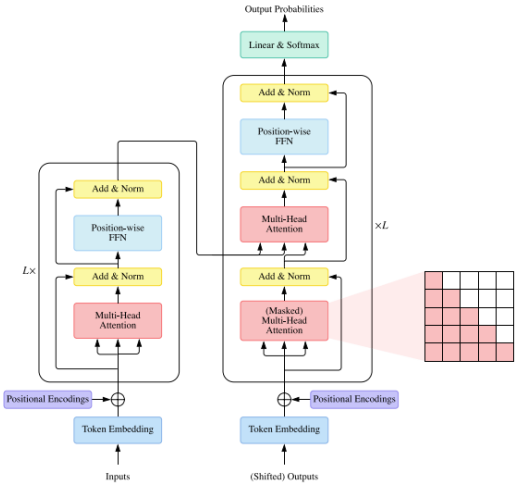

Figure 2.11: Overview of vanilla Transformer architecture

consists of an encoder and a decoder, each of which is a stack of L identical blocks. Each *encoder* block is mainly composed of a multi-head self-attention module and a position-wise FFN. For building a deeper model, a residual connection is employed around each module, followed by the Layer Normalization module. Compared to the encoder blocks, decoder blocks additionally insert cross-attention modules between the multi-head self-attention modules and the position-wise FFNs. Furthermore, the self-attention modules in the decoder are adapted to prevent each position from attending to subsequent positions. The overall architecture of the vanilla Transformer is depicted in Fig. 2.11.

2.4.3 Performance Metrics for supervised learning

2.4.3.1 Classification error

When it comes to intrusion detection, deep learning models may be evaluated using a variety of metrics. These measures evaluate several features of an IDS. For (supervised learning) classification methods, the primary metric from which all the other metrics are derived is the confusion matrix. We distinguish between two cases:

i) the confusion matrix of scalar binary label, i.e., $y_t \in \{0\ ,\ 1\}$

ii) the confusion matrix of D-dimensional binary label, i.e., $y_t \in \{0\ ,\ 1\}^D$

We define first the follow: True Positive (TP) is the number of intrusions correctly classified as an attack, True Negative (TN) is the number of normal instances correctly classified as

benign, False Negative (FN) is the number of intrusions incorrectly classified as benign, and False Positive (FP) is the number of normal instances incorrectly classified as an attack.

We start with the confusion matrix of a scalar label. The confusion matrix is a tabulation of classifications made by a model, typically with the actual class (ground truth) on rows and the predicted class on columns (displayed in Table 2.1 and Table 2.2). It shows the "classification distribution" of a model and helps identify the properties of the model, such as when it is constantly misclassifying one class as another.

		Predicted	
		Normal	Attack
Actual	Normal	TN	FP
	Attack	FN	TP

Table 2.1: Confusion matrix: scalar binary label

		Predicted		
		Class A	Class B	Class C
Actual	Class A	TP_A	E_{BA}	E_{CA}
	Class B	E_{AB}	TP_B	E_{CB}
	Class C	E_{AC}	E_{BC}	TP_C

Table 2.2: Confusion matrix for multiclass classification

Several other important metrics are calculated to measure the ability of an IDS to correctly identify and classify cyberattacks. Common metrics include accuracy (Acc), DR, false alarm rate (FAR), Precision, and F1 score. *Acc* shows the overall effectiveness of an algorithm. *DR*, also known as Recall, refers to the number of attacks detected among the total number of attack instances. Unlike *Recall*, *Precision* counts the number of attacks detected among the total number of instances classified as an attack. The F_1 *score* measures the harmonic mean of *Precision* and *Recall*. *FPR* is the number of normal instances classified as an attack divided by the total number of normal instances in the test dataset, while *FNR* shows the number of attack instances that are unable to be detected. Intuitively, the goal is to achieve a high *Acc*, *DR*, *Precision*, F_1 *score* and, at the same time, maintain low *FAR*.

These measures are defined by the following equations:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.16}$$

$$DR(Recall) = \frac{TP}{TP + FN} \tag{2.17}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.18}$$

$$FPR = \frac{FP}{TN + FP} \tag{2.19}$$

$$FNR = \frac{FN}{FN + TP} \tag{2.20}$$

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{2.21}$$

These metrics may be computed separately for each class in a multi-class problem. There are three possible techniques to average the per-class indicators to get the final measure.

- Micro average: calculate the metrics using the sum of each type of classification (eg: $TP = TP_{class1} + TP_{class2} + \cdots$
- Macro average: calculate the per-class metrics and average them (sum and divide by the no. of classes).
- Weighted macro average: similar to a macro average, except the per-class metrics are weighted by the number of instances in each class to obtain the final average.

It's noteworthy to mention that a natural trade-off exists between sensitivity (recall) and false positive rate (FPR). In fact, a highly sensitive model to attacks is likely to have a high false positive rate which is regarded as undesirable. The Receiver Operating Characteristic (ROC) curve illustrates this trade off. The FPR values are drawn on the horizontal axis and, sensitivity (recall) values are drawn on the vertical axis. The ROC curve is obtained by changing the threshold for classification into the positive class (attacks), and recording the corresponding FPR and recall. The specifics of the intended use case determine which ROC curve point must be selected. Moreover, Area Under Curve (AUC) is another metric for classifiers that reflects the ability of the model to distinguish between the two classes.

2.4.3.2 Computational Complexity

In addition to the metrics that evaluate the classification performance of a model, several other important metrics reflect model complexity by measuring the ability of an IDS to perform its tasks without being resource-intensive. Common computational metrics include:

- **Time to predict (inference time):** this can be measured as the time taken to make a prediction on a fixed-size batch of input instances. Since predictions in intrusion detection are made online (real-time), low latency figures are preferred.

- **Number of parameters:** this measure can help us infer the model size. In our case, small neural networks have a significant benefit as they are feasible to be deployed on ECUs and other in-vehicle hardware with limited memory. Additionally, small neural networks require less bandwidth to export a new model from the cloud to the vehicle.

Overall, the performance of an IDS can be evaluated by considering a combination of efficiency and computational metrics. A well-performing IDS should have a low false positive rate, a low miss detection rate, low processing time, and be low resource-intensive.

2.5 Conclusion

In this chapter, we provided a summary of the key concepts and the notation we use regarding in-vehicle networks, deep learning, and IDS, but it is not intended to be a representative overview. In the next chapters, we investigate the usefulness of different unsupervised and supervised neural network architectures when detecting and classifying attacks on various IVNs.

Part II

Controller Area Network

CHAPTER 3

CAN-IDS based on BERT Language Model

To protect modern vehicles against malicious attacks or malfunctions, it is crucial to detect anomalous messages transmitted in the CAN bus. In this chapter, we propose CAN-BERT, a self-supervised framework for CAN anomaly detection based on Bidirectional Encoder Representations from Transformers (BERT). CAN-BERT learns the patterns of normal CAN ID sequences by the novel masking self-supervised training task and is able to detect anomalies where the underlying patterns deviate from normal CAN ID sequences.

Contents

3.1 Introduction

To fulfill automotive features, the Controller Area Network (CAN) bus is widely used as the de-facto standard for message communication between different electronic control units (ECUs) in today's vehicles. It is sometimes referred to as a "message-based" system since it focuses on the transmission of diagnostic, informative, and controlling data through messages, also known as CAN data frames. In fact, while developing a vehicle, all conceivable CAN bus messages and their respective priority, encoded into an identifier called "CAN ID", must be determined beforehand.

Due to the lack of authentication, any device can connect physically or wirelessly to the CAN bus, and broadcast CAN data frames, and all the participants on the CAN bus can receive it without verifying its source. Consequently, since CAN security was not a design priority, many message injection attacks have become widely implemented. These attacks can interfere with the desired function of the system, shut down some connected nodes, and make the vehicle behave abnormally, putting at risk the safety of the driver and the passengers.

To address these security flaws, researchers have proposed intrusion detection as a supplementary layer of protection to specialized security solutions. By monitoring the communication between different ECUs within a CAN bus system, a network intrusion detection system (N-IDS) can detect deviations from the normal message exchange behavior and, thereby, identify both anticipated and novel cyberattacks. The adoption of deep neural networks for in-vehicle intrusion detection has lately proliferated, with impressive results. Since a message injection attack can alter the normal order of occurrence of several CAN IDs, researchers have deployed deep learning-based sequential models, to model the CAN ID sequences. Some studies have proposed the use of autoregressive models that are trained to capture the patterns of regular CAN ID sequences by predicting the future CAN ID sequence based on the preceding one, such as recurrent neural network (RNN) models and their variants and the generative pretrained transformer (GPT). However, these models identify malicious network intrusions on CAN ID traffic by focusing primarily on the exchange of CAN ID messages from earlier steps rather than integrating the left and right context of a CAN ID sequence, limiting the model's capacity to grasp the whole context information representation. Additionally, these algorithms focus largely on the correlation between CAN ID messages in normal sequences, which would result in false alarms for intrusion detection whenever the correlation is breached. Hence, due to these

limitations, the autoregressive models do not adequately depict the natural communication behavior between the various ECUs.

To address these challenges, we propose CAN-BERT, an intrusion detection system based on a language representation model called BERT. CAN-BERT, in contrast to autoregressive models, is a self-supervised model which can successfully depict deep bidirectional representations from CAN ID sequences by conditioning on both left and right context in its various layers. By using the "masked language model" unsupervised training objective, CAN-BERT model masks some of the CAN IDs in the input at random, with the goal of predicting the conventional ID of the masked word based on its left and right context.

We evaluate our approach using the recently published "Car Hacking: Attack & Defense Challenge 2020" collected from three different cars, Chevrolet Spark, Hyundai Sonata, and Kia Soul and which contain diverse types of message injection attacks.

Our contributions are summarized below:

- Inspired by the outstanding performance of BERT model for improving many natural language processing tasks, we propose "CAN-BERT", a novel BERT-based intrusion detection system architecture that can detect known and unprecedented cyberattacks in CAN ID sequences.

- We evaluate the performance of our approach with the recently published "Car Hacking: Attack & Defense Challenge 2020" collected from three different cars, Chevrolet Spark, Hyundai Sonata, and KIA Soul and which contain diverse types of message injection attacks. We also compare our model with other baseline models.

This chapter is structured as follows. In Section 3.2, we present the security weaknesses of the Controller Area Network (CAN) and the Bidirectional Encoder Representations from the Transformers model (BERT). In Section 3.3, we present an overview of our proposed framework "CAN-BERT". Section 3.4 discusses the launched experiments with the corresponding dataset. In Section 3.5, we discuss the obtained results showing the proposed model's accuracy and complexity.

3.2 Preliminaries

3.2.0.1 CAN's Security Weaknesses

CAN does not prohibit several ECUs from sharing the same IDs. Moreover, CAN messages are broadcast and do not contain any sender's address. Consequently, any device linked to the CAN bus can use any pre-defined ID, communicate its message without authentication or encryption, and all associated ECUs can receive it. The receiver defines whether or not a message identification causes the receiving ECU to retain and process the given data. Consequently, an attacker is able to broadcast spoofed CAN message, eavesdrop on all the CAN traffic, and collect detailed information about it, resulting in Fuzzing and Malfunction attacks.

Additionally, as previously mentioned, the CAN bus leverages the arbitration method which discerns between "dominant" (0) and "recessive" (1) bits in the message identifiers. Therefore, if several ECUs begin transmitting simultaneously, the ECU whose message begins with a greater number of dominant "0" bits will take over the CAN bus. As soon as a unit detects that the message on the bus is no longer the message it is transmitting, it halts its transmission, waits for the real transmission to conclude, then waits for the inter-frame gap to expire and retransmits its message. This phenomenon carries the risk that a message with a lower priority will never be delivered if the network is very congested and can be exploited by attackers to launch denial of service (DoS)/ flooding attacks.

3.2.1 BERT

Bidirectional Encoder Representations from Transformers (BERT), proposed by Devlin et al. [43], is a state-of-the-art language representation model which is designed to pretrain bidirectional representations from the unlabeled text by jointly conditioning on both left and right context in all layers. Regarding its architecture, it is a multi-layer bidirectional Transformer encoder based on the original implementation proposed by Vaswani et al. [127].

Inspired by its outstanding performance in modeling sequential data, BERT is recently employed for sequence anomaly detection [90] [52] [88] [85] [140]. To the best of our knowledge, none of the previous works have evaluated the performance of the BERT model for in-vehicle intrusion detection on CAN protocol.

In order to detect anomalies in sequences, it's crucial to incorporate context from both left and right directions of the sequence. Sequential anomalies may be misdetected by traditional unidirectional models, such as OpenAI GPT and RNNs, where every token can only attend to

context to its left. To solve this significant constraint, some researchers have proposed a shallow concatenation of both left-to-right and right-to-left architecture of the autoregressive models, such as Bi-RNN and Bi-GPT [104]. However, these approaches aren't as powerful as BERT which adopts a "masked language model" (MLM) training objective, in which input sequence tokens are randomly masked and the goal is to predict the original vocabulary id of the masked word based on its context. In contrast to denoising auto-encoders, BERT predicts the masked words instead of reconstructing the whole sequence [129].

3.3 Proposed framework: CAN-BERT

We propose "CAN-BERT", a pattern-based anomaly detection algorithm, which leverages a BERT-based architecture to detect message injection intrusions in the CAN bus. As seen in Figure. 3.1, our model is composed of a multi-layer bidirectional Transformer encoder and is trained using the "masked language model" self-supervised task to model normal CAN ID sequences. The following subsections elaborately describe the suggested framework.

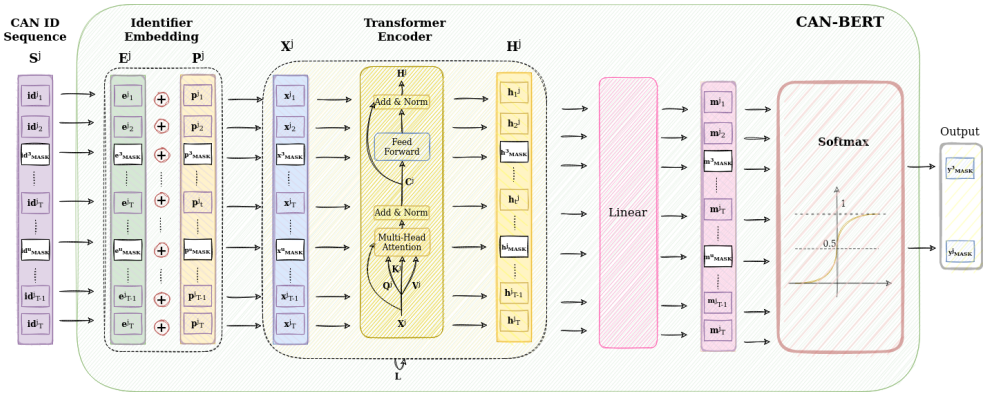

Figure 3.1: CAN-BERT model architecture

3.3.1 Model description

Note that $\mathbf{S} = [\mathbf{id}_1, ..., \mathbf{id}_t, ..., \mathbf{id}_T]$ is an observed sequence of T CAN identifiers, arranged in their order of transmission in the CAN bus network, where an identifier $\mathbf{id}_t \in \mathbb{ID}$ is an M-dimensional vector which denotes the CAN ID transmitted at time t by an ECU, $\mathbb{ID}$ indicates the set of CAN IDs extracted from CAN message, and M is the size of the $\mathbb{ID}$ set.

Since anomaly detection is an unsupervised learning-based technique in which only normal data are used for training, a collection of N CAN ID sequences, represented as $\mathbb{D}_{training} = \{\mathbf{S}_1, ..., \mathbf{S}_j, ..., \mathbf{S}_N\}$, is solely used as a training dataset.

Identifier Embeddings To feed the appropriate input to the BERT model, each identifier $\mathbf{id}_t^j$ with size (M,1) in a CAN sequence $\mathbf{S}^j$ is firstly projected into a d-dimensional space using a single linear layer, i.e.:

$$\mathbf{e_t^j} = \mathbf{W}^e \mathbf{id_t^j} + \mathbf{b}^e, \forall i \in \{1..T\}, \forall j \in \{1..N\} \tag{3.1}$$

where $\mathbf{e_t^j}$ represents the identifier embedding with size (d,1), $\mathbf{W}^e \in \mathbb{R}^{d \times M}$ is the input embedding weight matrix, and $\mathbf{b}^e \in \mathbb{R}^d$ denotes the bias. Both $\mathbf{W}^e$ and $\mathbf{b}^e$ are trainable parameters.

Subsequently, the identifier's position is encoded into a d-dimensional positional embedding $\mathbf{p_t^j}$ using a sinusoidal function. To this end, the CAN ID, fed into the CAN-BERT model, is a summation of both the positional encoding and the input embedding :

$$\mathbf{x}_t^j = \mathbf{e_t^j} + \mathbf{p_t^j} \tag{3.2}$$

where $\mathbf{x}_t^j$ is the total embedding j-th identifier in the t-th CAN ID sequence $\mathbf{id}_t^j$, thereby the convergence of the input sequence $\mathbf{S}^j$ into $\mathbf{X}^j = [\mathbf{x}_1^j, ..\mathbf{x}_t^j .., \mathbf{x}_T^j]^T$ with $\mathbf{X}^j$ a matrix with size $T \times d$.

Transformer Encoder The encoded input $\mathbf{X}^j$ is then delivered into a stack of L transformer encoder layers, each of which has two sub-layers: a multi-head self-attention mechanism and a position-wise feed-forward network [127]. A residual connection is employed around each of these two sub-layers, followed by layer normalization [29], as follows:

$$\begin{aligned} \mathbf{E}^{(j,l)} &= g(\mathbf{X}^{(j,l)}) + f(\mathbf{X}^{(j,l)} + g(\mathbf{X}^{(j,l)})) \\ \mathbf{H}^{(j,l)} &= z(\mathbf{E}^{(j,l)}) + f(\mathbf{E}^{(j,l)} + z(\mathbf{E}^{(j,l)})) \\ \mathbf{X}^{(j,l+1)} &= \mathbf{H}^{(j,l)}, \forall l < L \end{aligned} \tag{3.3}$$

where $\mathbf{E}^{(j,l)}$ represents the output of the first sublayer for the l-th transformer encoder layer with size $T \times d$, $\mathbf{H}^{(j,l)}$ represents the output of the second sublayer for the l-th transformer encoder layer with size $T \times d$, g is the multi-head attention function, z is the position wise feed forward function, and f is the layer normalization function.

Attention We use the scaled dot-product attention proposed by [127], requiring query $\mathbf{Q}^{(j,l)}$, key $\mathbf{K}^{(j,l)}$, and value $\mathbf{V}^{(j,l)}$ representations, and which are projections of the embedded sequence $\mathbf{X}^{(j,l)} \in \mathbb{R}^{T \times d}$. In fact, we leverage the dot-product similarity to compare the query

representation of a given CAN identifier to all other keys. Hence, if the query and key are comparable and have a high attention weight, the matching value is deemed to be relevant. The output is therefore computed as a weighted sum of the values $\mathbf{V}$:

$$\begin{aligned} Attn(\mathbf{Q}^{(j,l)}, \mathbf{K}^{(j,l)}, \mathbf{V}^{(j,l)}) &= \sigma(\frac{\mathbf{Q}^{(j,l)}\mathbf{K}^{(j,l)T}}{\sqrt{d}})\mathbf{V}^{(\mathbf{j,l})} \\ Attn(\mathbf{Q}^{(j,l)}, \mathbf{K}^{(j,l)}, \mathbf{V}^{(j,l)}) &= \mathbf{A}\mathbf{V}^{(j,l)} \end{aligned} \tag{3.4}$$

where σ is the softmax function, $\mathbf{A} \in \mathbb{R}^{T\times T}$ denotes the attention weight matrix containing attention weights, and d is the dimension of the $\mathbf{Q}^{(j,l)}$, $\mathbf{K}^{(j,l)}$,$\mathbf{V}^{(j,l)}$ vectors.

As described by [127], the multiple heads of attention allows the model to concurrently capture diverse aspect of data at distinct CAN IDs. Hence, we adopt a multi-head attention (MHA) mechanism in which the d-dimensional CAN identifers are projected into subspaces calculated by different attention heads $n \in \{1, .., H\}$:

$$\begin{aligned} \mathbf{Q}^{(j,n,l)} &= \mathbf{X}^{(j,l)}\mathbf{W}^{(Q,n)}, \mathbf{Q}^{(j,n,l)} \in \mathbb{R}^{T\times F} \\ \mathbf{K}^{(j,n,l)} &= \mathbf{X}^{(j,l)}\mathbf{W}^{(K,n)}, \mathbf{K}^{(j,n,l)} \in \mathbb{R}^{T\times F} \\ \mathbf{V}^{(j,n,l)} &= \mathbf{X}^{(j,l)}\mathbf{W}^{(V,n)}, \mathbf{V}^{(j,n,l)} \in \mathbb{R}^{T\times F} \end{aligned} \tag{3.5}$$

where $\mathbf{Q}^{(j,n,l)}$, $\mathbf{K}^{(j,n,l)}$ and $\mathbf{V}^{(j,n,l)}$ are the query, key and value vectors, respectively of the $j-th$ CAN ID sequence for the l-th transformer encoder layer and which are calculated using the n-th attention head. The $\mathbf{W}^{(Q,n)}$, $\mathbf{W}^{(K,n)}$ and $\mathbf{W}^{(V,n)}$ are their corresponding trainable weight matrices $\in \mathbb{R}^{d\times F}$, and F is set to D/H. The results are then concatenated and projected back into representation space using the weight matrix $\mathbf{W}^{o} \in \mathbb{R}^{HF\times D}$ as follows:

$$\mathbf{head}_n^{(j,l)} = Attn(\mathbf{Q}^{(j,n,l)}, \mathbf{K}^{(j,n,l)}, \mathbf{V}^{(j,n,l)}) \tag{3.6}$$

$$\overline{\mathbf{X}}^{(j,l)} = [\mathbf{head}_1^{(j,l)}, ..\mathbf{head}_n^{(j,l)}, .., \mathbf{head}_H^{(j,l)}]\mathbf{W}^{O} \tag{3.7}$$

where $\overline{\mathbf{X}}^{(j,l)} \in \mathbb{R}^{(T,d)}$.

Position-wise feed-forward A position-wise feed-forward network with a ReLU activation is thereby applied to each representation in each of the layers of our encoder, in addition to attention sub-layers, using the following equation:

$$z(\mathbf{E}^{(j,l)}) = [\mathbf{W_1}\mathbf{E}^{(j,l)}]_+ \circ \mathbf{W_2} \tag{3.8}$$

where $\mathbf{E}^{(j,l)}$ is previously defined in (3.3), $\mathbf{W}_1$ and $\mathbf{W}_2$ are trainable projection matrices, where $\circ$ is the hadarmard product, and $[]_+$ is the element-wise maximum.

After passing through different transformer layers, the L-th contextual embedding vectors of the CAN IDs, denoted as $\mathbf{h}_t^{(j,L)}$ with size $(d,1) \in \mathbf{H}^{(j,L)} = [\mathbf{h}_1^{(j,L)}, .., \mathbf{h}_T^{(j,L)}]^T$, are fed into a single linear layer which projects them back to the M-dimensional layer, as follows:

$$\mathbf{m_t^j} = \mathbf{W}^m \mathbf{h}_t^{(j,L)} + \mathbf{b}^m, \forall i \in \{1..T\}, \forall j \in \{1..N\} \tag{3.9}$$

where $\mathbf{m_t^j}$ represents the projected output with size (M,1), $\mathbf{W}^m \in \mathbb{R}^{M \times d}$ is the input embedding weight matrix, and $\mathbf{b}^e \in \mathbb{R}^M$ denotes the bias. Both $\mathbf{W}^m$ and $\mathbf{b}^m$ are trainable parameters.

3.3.2 Training and Inference

We use the masked language model training method to train the CAN-BERT model on capturing the patterns of normal CAN ID sequences. Hence, CAN sequences with random mask as inputs, where we randomly replace a ratio of CAN IDs in a sequence with a specific MASK token, are fed into CAN-BERT. The purpose of training is to reliably anticipate the CAN IDs that have been randomly masked.

To achieve that, we feed the contextual embedding vector of the u-th MASK in the j-th CAN ID sequence $\mathbf{m}^j_{MASK_u}$ to a softmax function, which will return a probability distribution for the whole set of CAN IDs $\mathbb{ID}$:

$$\widehat{\mathbf{y}}^j_{[MASK_u]} = \sigma(\mathbf{W}^c \mathbf{m}^j_{[MASK_u]} + \mathbf{b}^c) \tag{3.10}$$

where $\widehat{\mathbf{y}}^j_{[MASK_u]}$ is an m-dimensional vector, σ is the softmax function, $\mathbf{m}^j_{[MASK_u]}$ and $\mathbf{b}^c$ are trainable parameters.

CAN-BERT is trained to minimize the cross entropy loss over a batch of I sequences (with $I \leq N$), for masked CAN ID prediction, which is defined as:

$$\mathcal{L}_{MASK} = -\frac{1}{IR} \sum_{j=1}^{N} \sum_{u=1}^{R} \mathbf{y}^j_{[MASK_u]} log \widehat{\mathbf{y}}^j_{[MASK_u]} \tag{3.11}$$

where the ground-truth u-th CAN ID in the j-th sequence is denoted as $\mathbf{y}^j_{[MASK_u]}$, R is the total number of masked tokens in the j-th sequence, and N is the number of training samples.

By modeling the normal exchange of messages through CAN bus using CAN-BERT, our model is expected, after training, to predict a candidate set with the normal CAN IDs having the highest likelihood for each masked token. Hence, for a randomly masked testing sequence, we calculate the probability distribution represented in (3.10), for each masked CAN ID. Therefore, if the actual CAN ID is among the anticipated candidates, the corresponding CAN ID sequence is considered as normal. Otherwise, it is deemed abnormal.

3.4 Experimental Settings

3.4.1 Dataset

To assess the proposed CAN-BERT, we leverage the “In-Vehicle Network Intrusion Detection Challenge" dataset [77] (presented in Table 3.2), which was used at the “In-vehicle Network Intrusion Detection track’ of ‘Information Security R&D Data Challenge 2019.

Vehicle	Dataset	# Normal packets	# Abnormal packets	Size (MB)
CHEVROLET Spark	Attack Free	136,933	N/A	6.2
	Flooding	70,001	14,999	4.2
	Fuzzy	37,957	3,043	2.0
	Malfunction	47,005	3,995	2.5
HYUNDAI Sonata	Attack Free	117,172	N/A	5.8
	Flooding	78,907	17,093	4.9
	Fuzzy	78,905	9,095	4.5
	Malfunction	78,798	8,202	4.5
KIA Soul	Attack Free	192,515	N/A	9.3
	Flooding	103,928	16,072	6.2
	Fuzzy	122,387	21,613	7.4
	Malfunction	108,230	4,770	5.8

Table 3.1: In-Vehicle Network Intrusion Detection Dataset

The dataset includes normal and abnormal in-vehicle network traffic data of HYUNDAI Sonata, KIA Soul, and CHEVROLET Spark vehicles collected during their stationary state. We have mainly used its preliminary dataset, which includes three types of attacks (Flooding, Fuzzy, and Malfunction). The dataset is labeled and each sample is represented by the following features: “Timestamp" representing the logging time, “CAN ID" representing the CAN Identifier, “DLC" indicating the Data length code, and the Payload indicating the “CAN data" field.

3.4.1.1 Attacks

The dataset contains the following attacks:

- **Flooding Attack** The flooding attack was carried out by injecting many messages with the CAN ID set to 0x000 into the CAN network. Consequently, an ECU that transmits CAN data frames with such CAN ID dominates the CAN bus, which could restrict the communications among the ECU nodes and impair normal in-vehicle functions.

- **Fuzzy Attack** To implement fuzzy attacks, the attacker injected every 0.0003 seconds random CAN packets, for both the ID field and the Data field. This process lead to

abnormal automotive functionalities behavior such as short beeping sound repeatedly occurring, the heater turning on, etc.

- **Malfunction Attack** The malfunction attack was carried out by injecting messages with a specified CAN ID from among the extractable CAN IDs of a particular vehicle in order to disable relevant automotive functions, such as IDs 0×316, 0×153 and 0×18E for the HYUNDAI YF Sonata, KIA Soul, and CHEVROLET Spark vehicles, respectively.

$$c(\mathbf{x}) = \begin{cases} 0 \text{ (normal)} & s_x < \beta \\ 1 \text{ (abnormal)} & s_x > \beta \end{cases} \tag{3.12}$$

where c($\mathbf{x}$) is the classification function for input sample $\mathbf{x}$ and β is the pre-defined anomaly detection threshold.

Vehicle	Dataset	# Normal packets	# Abnormal packets	Size (MB)
CHEVROLET Spark	Attack Free	136,933	N/A	6.2
	Flooding	70,001	14,999	4.2
	Fuzzy	37,957	3,043	2.0
	Malfunction	47,005	3,995	2.5
HYUNDAI Sonata	Attack Free	117,172	N/A	5.8
	Flooding	78,907	17,093	4.9
	Fuzzy	78,905	9,095	4.5
	Malfunction	78,798	8,202	4.5
KIA Soul	Attack Free	192,515	N/A	9.3
	Flooding	103,928	16,072	6.2
	Fuzzy	122,387	21,613	7.4
	Malfunction	108,230	4,770	5.8

Table 3.2: In-Vehicle Network Intrusion Detection Dataset

As mentioned in Section 3.3, we aim to detect if a sequence of ordered CAN ID contains injected messages. Hence, in order to represent CAN ID sequences, we use the **Feature-based Sliding Window (FSW)** to group CAN IDs which belong to the dataset into subsequences with fixed window size $\mathbf{T}$, where $\mathbf{T} \in \{16, 32, 64, 128, 256\}$ and the slide size is 1. Moreover, each CAN ID sequence $\mathbf{S} = [\mathbf{id}_1, ..., \mathbf{id}_t, ..., \mathbf{id}_T]$ has its corresponding labels represented by $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_t, ..., \mathbf{y}_T]$ wherein each identifier $\mathbf{id}_t \in \mathbf{S}$ is labeled as $\mathbf{y}_t = 1$ if $\mathbf{id}_t$ is an injected identifier in $\mathbf{S}$ or as $\mathbf{y}_t = 0$ otherwise. However, to identify the state of each sequence, we have used the following criteria:

$$z = \begin{cases} 1 \text{ (abnormal)} & \text{if } \exists \mathbf{y}_t = 1, \forall t \in \{1, .., T\} \\ 0 \text{ (normal)} & \text{otherwise} \end{cases}$$

where z is the CAN ID sequence's label.

3.4.2 Benchmark Models

We chose the PCA, IF, and autoencoder models (described in Chapter 2), which are regarded as the benchmark models for evaluating the performance of different CAN ID sequences. For the autoencoder model, we have tested Long short-term based memory (LSTM) and Bidirectional LSTM (BiLSTM)-based models with different network hyperparameters: BiLSTM-AE-4 (with 4 layers), LSTM-AE-4 (with 4 layers), and LSTM-AE-8 (with 8 layers).

3.5 Results

For measuring the performance of different anomaly-based IDS, we use the F1-score metric described in Chapter 2. To evaluate our model, we leverage the Python deep learning framework Pytorch [13]. We train and evaluate them on NVIDIA® Tesla® V100S with 32 GB HBM2 memory.

3.5.1 Model Configuration & Hyperparameter tuning

Parameter	Value
N	4
d_{model}	256
d_{ff}	512
h	1
P_{drop}	0.1
m	0.45
# Candidates	5
Optimizer	Adam
Adam β_1	0.9
Adam β_2	0.999
Learning rate	0.001
Batch size	32
# Epochs	200
Patience	10

Table 3.3: CAN-BERT model configuration

As presented in Table 3.3, for CAN-BERT, we have chosen the total number of transformer encoder layers as 4. In each transformer layer, the position-wise feed-forward network is composed of two dense layers where the first one projects d=256 dimensional CAN identifier embedding into d_{ff}= 512-dimensional space, followed by a ReLU activation. The second dense layer maps back the 512-dimensional vector into the d-dimensional space.

For training, we use a batch size of 32, a learning rate of 0.001, and the Adam optimizer [80] with its default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. To avoid overfitting, we employ the same dropout of P_{drop}=0.1 for all dropout layers in our network. Moreover, we apply early stopping for a total number of 200 epochs and a patience of 10 epochs as a form of regularization used to avoid overfitting.

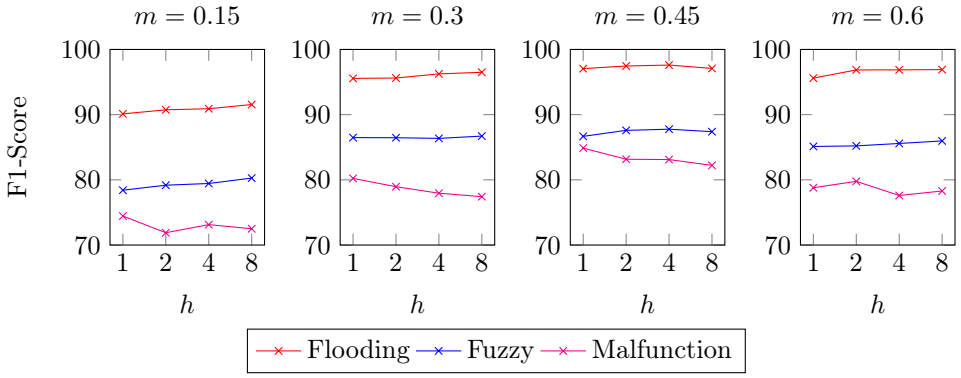

Figure 3.2: Hyperparameter tuning the mask ratio m and the number of attention heads h on the "CHEVROLET Spark" dataset for T=32. We have obtained similar behavior pattern for the results w.r.t other sequence length and car types.

The hyperparameters, including the ratio of masked CAN IDs in a sequence m, and h the number of attention heads are tuned based on a validation set for the three car types and the different message injection attacks. As seen in Figure 3.2, raising the ratios of masked CAN IDs in the sequences from 0.1 to 0.45 somewhat improves F1 scores, however increasing the ratios further degrades performance, as is the case for m=0.65. Furthermore, the model performance is relatively stable by setting different attention head $h \in \{1, 2, 4, 8\}$ values for each mask ratio $m \in \{0.15, 0.3, 0.45, 0.6\}$. Therefore, in our in-vehicle intrusion detection use case, a single attention head is sufficiant to detect different types of intrusions. Note that, in our experiments, we use the same ratio of masked CAN IDS m=0.45 and h=1 for both training and inference phases.

3.5.2 Model Accuracy

As seen in Figure 3.3, we compare performance of the CAN-BERT model with other baselines approaches for different sequence length T using the F1-score metric. In fact, we varied the sequence length among values of 16, 32, 64, 128 and 256 CAN IDs per sequence in the experiments. If our approaches could identify a message injection attack in a shorter sequence

length, it would be more advantageous in a practical situation. The traditional machine learning

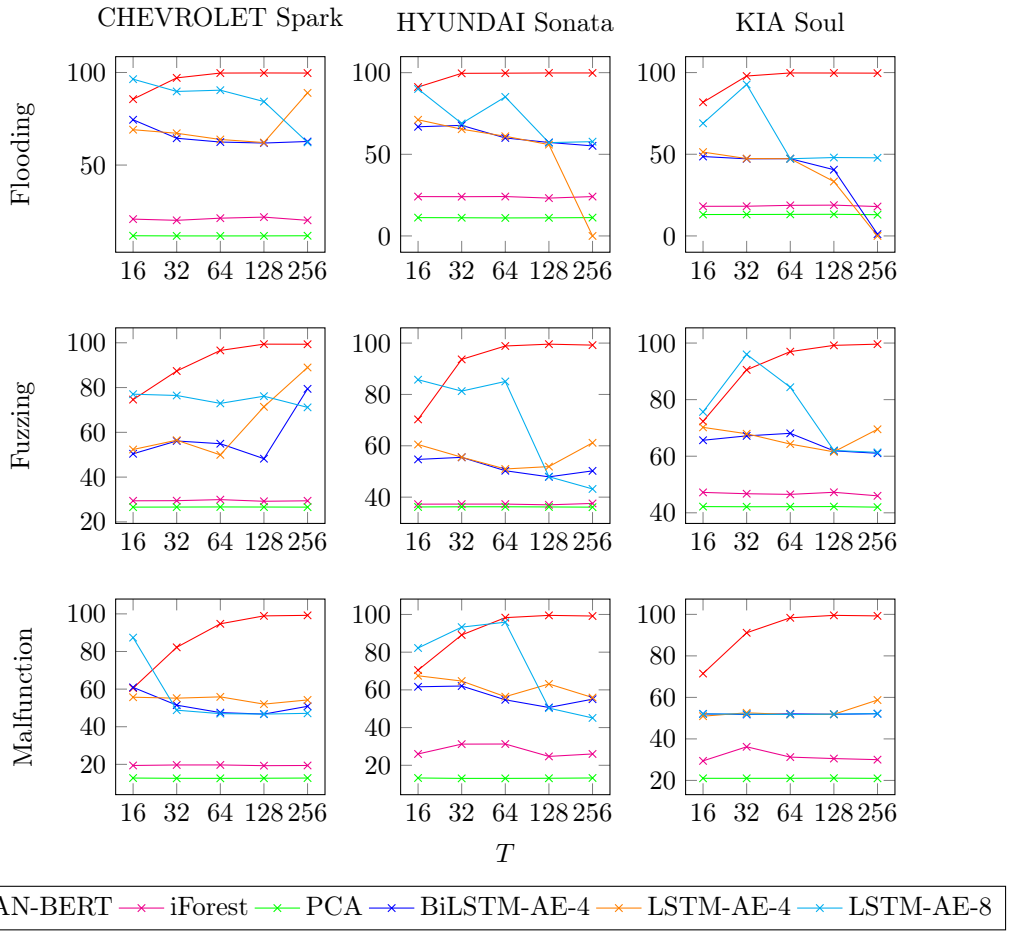

Figure 3.3: Comparision of the CAN-BERT model with other anomaly detection baselines using the F1-score percentage metric, for different message injection attacks applied on different car models w.r.t sequence length T.

algorithms such as Isolation Forest (iForest), and Principal Component Analysis (PCA) perform poorly and maintain the same F1-score metric w.r.t sequence length. Because these models presume small datasets with a limited number of features, they fail to discover abnormalities in high-dimensional datasets. Because of this, a significant fraction of irrelevant features may effectively disguise the underlying abnormalities in the input data, resulting in poor anomaly identification performance when dealing with large input dimensions. Meanwhile, both deep learning-based models autoencoder (AE) and CAN-BERT outperformed the traditional anomaly detection models over different window sizes. However, when the length of the CAN ID sequence is increased, both models performed oppositely. In contrast to the baseline models, our suggested

model, CAN-BERT, significantly outperforms them by huge margins and obtains respectable F1 scores $\in [0.85, 0.99]$, demonstrating the usefulness of using BERT-based models to capture the patterns of CAN ID sequences when $T \geq 32$. However, on short sequence length, as is the case for $T = 16$, the model performs modestly with F1-score $\in [0.6, 0.9]$ for different kinds of attacks. These experiments, therefore, reveal that by using self-supervised training tasks, CAN-BERT can effectively model medium to long normal CAN ID sequences and accurately detect anomalous sequences.

3.5.3 Model Complexity

From a practical point of view, we must assess not only the classification performance but also the model complexity to check if the model's ability for real-time in-vehicle intrusion detection in CAN networks. Therefore, we assessed the inference time per sample as well as the number of parameters for the CAN-BERT model w.r.t different car types.

Vehicle	Features	Values
CHEVROLET Spark	Number of Parameters	2,937,422
	Model Size (MB)	$[20, 70]$
	Inference Time (ms)	$[0.8, 3.1]$
HYUNDAI Sonata	Number of Parameters	3,149,291
	Model Size (MB)	$[20, 70]$
	Inference Time (ms)	$[0.8, 3.5]$
KIA Soul	Number of Parameters	3,163,142
	Model Size (MB)	$[20, 70]$
	Inference Time (ms)	$[0.8, 3.8]$

Table 3.4: CAN-BERT model complexity

As depicted in Table 3.4, the intrusion detection inference time varies between 0.8 and 3.1 ms w.r.t CAN ID sequence length. Hence, when considering a sequence length of 32 CAN IDs, our model detects an intrusion in 0.9 to 1 ms, which is suitable for real-time detection. Furthermore, having a size between 20MB and 70 MB and a number of parameters ranging between 2 to 3 million, our model can be either deployed in a performant ECU or even on a cloud server wirelessly connected to the vehicle.

3.6 Conclusion

We proposed CAN-BERT, a self-supervised intrusion detection system based on the BERT model, for in-vehicle intrusion detection and is, therefore, able to capture the contextual relations between CAN IDs. The strength of the proposed model has been demonstrated for different

sequence lengths of CAN messages and by its ability to surpass state-of-the-art techniques for efficient real-time detection. Moreover, as previously discussed, the introduction of Automotive Ethernet as an IVN does not indicate that it completely replaces existing IVNs, particularly the CAN bus. In fact, the aim of Automotive Ethernet is to unify the in-vehicle communication network technology. Since CAN protocol will certainly be used in Automotive Ethernet networks for specific automotive functionalities, the proposed BERT-based model must also be evaluated for representative datasets of the heterogeneous networks. Moreover, it would be interesting to see if the intrusion detection performance of the BERT model could be enhanced by correlating security events across many diverse IVN protocols [147]. Han et al. [55] have recently proposed a framework that detects and identifies corresponding abnormalities based on wavelet transform and customized DCNN. Three protocols were covered including AVTP, generalized Precision Time Protocol (gPTP) [1], and CAN protocol. As they have recently published their dataset, we would leave our research question open.

In the next chapter, we assess the performance of BERT-based models in detecting timing opaque attacks,i.e., attacks that do not affect the normal sequencing of CAN ID but rather alter the carried signal values. We thus develop a model which predicts stealthier intrusions that affect interdependencies between several CAN signals transmitted by distinct IDs.

[1] gPTP performs precision, time synchronization of all endpoints included in the AVB network and provides the time information required for audio or video stream transmission

CHAPTER 4

BERT-based multi-agent IDS for CAN

Predicting sophisticated intrusions that affect interdependencies between several CAN signals transmitted by distinct IDs requires modeling the temporal relationships between signals carried by each ID separately along with the interaction between CAN IDs. In this chapter, we propose a novel deep learning-based multi-agent intrusion detection system, AMICA, that uses an attention-based self-supervised learning technique to detect stealthy in-vehicle intrusions, i.e., those that not only disturb normal timing or ID distributions but also carried data values by multiple IDs, along with others.

Contents

4.1 Introduction

Despite being widely deployed in cars today, the Controller Area Network (CAN) technology, responsible for delivering reliable communication between several ECUs lacks important security mechanisms mainly authentication and encryption. To mitigate the severe aftermath, numerous intrusion detection approaches have been proposed to detect attacks against IVN. In such settings, observations of CAN network packets are sent sequentially to a detector that is tasked with detecting threats, particularly those that may not be caught by other security measures, such as zero-day vulnerabilities or insider attacks.

Unfortunately, most available approaches for intrusion detection don't handle the challenging structure of the CAN protocol. A suitable IDS for CAN must take into consideration its challenging message transmission mechanism, i.e only a single message can be transmitted at any point in time. In fact, some intrusions can only be detected by monitoring the interdependencies of several transmitted signals asynchronously. To overcome the shortcomings of previous approaches in the context of CAN IDS, we present **AMICA**, an attention-based IDS, that monitors asynchronously transmitted signals carried by several CAN Identifiers (ID) in the CAN. Inspired by the success of BERT model in detecting in-vehicle intrusions and the outstanding performance of Transformers in addressing multi-agent problems, we propose a BERT-based IDS that predicts intrusions on CAN by modeling two key dimensions:

1. ***time dimension***, where we model the temporal relationships between signals carried by each ID separately

2. ***interaction dimension*** where we model the interaction between IDs, i.e., how the state of each CAN ID affects the others.

4.2 Related Work

Techniques based on unsupervised deep learning have been widely used for detecting known and unknown intrusions on the CAN bus [96, 106]. Despite their efficiency, most of these works consider detecting intrusions on signals transmitted by each ID separately without taking into consideration their interdependencies, thus incapable of detecting sophisticated attacks. Few works have addressed this challenging problem such as [59] who introduced separate LSTM modules for each CAN ID followed by a shared autoencoder-based module "CANet" tailored to work on the signal space of CAN data. However, there are still some limitations to using LSTM

for modeling sequential data. First, although LSTM can capture the sequential information by the recurrence formula, it cannot make each element in a sequence encoding the context information from both the left and right context. However, it is crucial to observe the complete context information instead of only the information from previous steps when detecting malicious attacks based on CAN messages.

To address the existing limitations of LSTM-based models, researchers have started to leverage the Bidirectional Encoder Representations from Transformers (BERT) [43] for anomaly detection. Recently, Alkhatib et al. [27] proposed CAN-BERT, an IDS based on the BERT model tasked with monitoring the sequence of transmitted IDs through time and detecting intrusions on CAN bus by explicitly encoding the common patterns shared by all CAN ID normal sequences. However, their proposed model only regards the timing of each ID and/or the sequential nature of IDs and cannot detect attacks that do not disrupt normal timing or ID distributions.

Devising a multi-agent intrusion detection system is challenging since the interrelations between asynchronous CAN signals transmitted by distinct agents (IDs) are complex. To overcome the challenges of building such systems, Transformers have been widely adopted [127]. In fact, Arnab et al. [28] and Gedas et al. [33] proposed a spatio-temporal model for video classification tasks. Achaji et al. [25] and Ye et al. [141] proposed a transformer-based multi-agent model that takes into account the interactions between several agents with respect to time and space for multi-agent trajectory prediction.

4.3 Proposed framework: AMICA

In this section, we describe our proposed model, AMICA, which tackles the asynchronous nature of the CAN protocol, in which messages are sent by various identifiers at different times. The different elements of the model are described as follows:

4.3.1 Input Formulation

Let $I = \{A_1, A_2, \ldots, A_N\}$ be the set of all possible CAN IDs. The input to the model, denoted by X, is composed of all the messages sent by different IDs in a temporal window of horizon T. We define the input as $X = \{X_1, X_2, \ldots, X_N\}$, where $X_i = \{x_{i,t_1}, \ldots, x_{i,t_j}, \ldots, x_{i,t_M}\}$ represents A_i's ordered set of message payloads, i.e. signals, transmitted during the horizon T, $t_j < T$ is the message global timestamp with respect to T, and N is the total number of IDs.

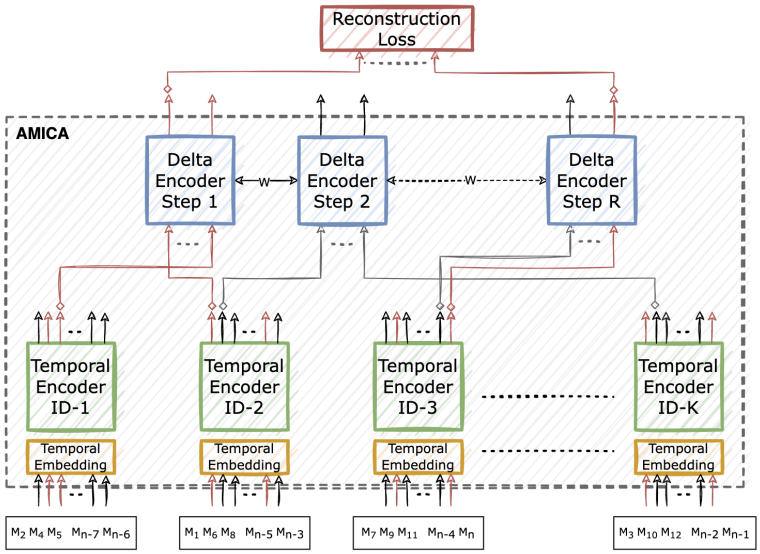

Figure 4.1: Overview of the AMICA model architecture. A sequence of ordered CAN messages (signals) M_t are fed to their corresponding temporal ID encoder. In each sequence, a pre-defined ratio of the input messages is masked (colored in red). After being processed by their corresponding temporal ID encoder, the Delta encoder receives them and outputs the encodings of the previously masked messages in a fixed time window.

4.3.2 Temporal Module

4.3.2.1 Temporal Embedding

Message $x_{i,t} \in R^{d_i}$ transmitted at time t and carried by ID A_i is fed into its corresponding Temporal-based module TM_i. Each module TM_i has a different set of weights that will be optimized separately. Since different IDs can transmit various amounts of signals, all messages will be projected into a $d-$dimensional space via a single linear layer where $x_{i,t} = W_i x_{i,t} + b_i$. $x_{i,t}$ represents the projected signal, $W_i \in R^{d \times d_i}$ are the weights, and $b_i \in R^d$ represents the bias. Transformer models have no notion of time when computing attention for each of the input's elements. Usually, Transformer embedding layers are coupled with a positional encoding layer that will inject the timestamp of each input message $x_{i,t}$ when fed to its associated module TM_i. However, since messages are sent through asynchronous transmission, the input X composed of ordered X_i sets is an unordered set of $x_{i,t}$ signals. Thus, we employ a global positioning encoding layer that will encode the relative position of a message $x_{i,t}$ with respect to the input set X. The global positional encoding will follow the same sinusoidal formulation as in [**?**] and will be a function of the global timestamp t, $\tau(t)$. As opposed to the local positional encoding,

a global positional encoding layer can be helpful to make the model learn the relative temporal dependencies between signals transmitted by different CAN IDs. The temporal embedding $x_{i,t}$ will be aggregated by $\tau(t)$, $x_{i,t} = x_{i,t} + \tau(t)$.

4.3.2.2 Temporal Encoder

The Temporal encoder is a bidirectional model that uses the scaled dot-product self-attention layer proposed by [1]. Unlike, LSTM-based solutions [59] that leverage left-to-right temporal conditioning, bidirectional models are strictly more powerful [43]. The self-attention layer is an operation over the queries (**Q**), keys (**K**), and values (**V**) vectors:

Attention(Q,K,V) = softmax($\mathrm{QK}^T/\sqrt{D} + M)V = AV$

Q, **K**, and **V** are the parametric linear projections of the input embedding vector X_i. In the Temporal Encoder case, the attention weights A denote the relative score given to each time step compared to the other time steps for each ID A_i. Since the number of transmitted messages carried by each ID is dynamic over time, we padded the input sequence to a fixed number of messages and then applied a corresponding padding mask M to the $Softmax$ function presented in (4.3.2.2). In addition to the attention layer, the temporal encoder is also composed of a point-wise feed-forward (Pff) and normalization layers (LN) presented by the following equations:

$$X_i = LN(X_i + Attention(X_i, X_i, X_i)) \tag{4.1}$$

$$X_i = LN(Pff(X_i) + X_i) \tag{4.2}$$

4.3.3 Interaction Module

The interaction module is responsible for encoding the inter-dependencies of signals carried by different IDs. In contrast to the spatio-temporal and multi-agent systems [25, 28], the signals are transmitted asynchronously in the CAN bus, i.e., it is not possible to catch the correlations between signals with different IDs at the same time. Thus, we build an encoder that takes as input an ordered set of signals, denoted as X_Δ, and which correspond to a set of encoded signals with different IDs in a range of horizon ΔT, where ΔT is a hyperparameter $\in [1, T]$. The ratio $R_\Delta = T/\Delta T$ represents a trade-off between the model interaction capabilities and the model complexity. For instance, $R_\Delta = 1$ means that the model can calculate a correlation between all signals transmitted during **T** timestamps. However, since the complexity of attention-based modules is quadratic in terms of sequence length, increasing R_Δ can decrease the model

complexity from $c = O(T^2)$ in the ultimate case to $c = O(R_\Delta \cdot \Delta T^2) = O(T \cdot \Delta T) \leq O(T^2)$ in general cases.

4.3.3.1 Delta Embedding

Similar to the temporal embedding in Section (4.3.2.1), we have applied temporal encoding with the purpose of injecting a notion of time to the later attention modules. However, the temporal encoding in this module is a local temporal encoding $\tau(p)$ representing the relative position p of a signal w.r.t the local X_Δ input. Additionally, a second encoding that reflects the ID encoding is implemented in this module. It injects an encoding $\tau(i)$ based on the message carried by the A_i for each message in X_Δ. The output embedding will be formulated as $x_{\Delta_{i,p}} = W x_{\Delta_{i,p}} + \tau(i) + \tau(p)$, where W are the weights of the linear projection layer.

4.3.3.2 Delta Encoder

The delta encoder has the same architecture and associated equations of the temporal encoder presented in section (4.3.2.2) following equations (4.1), (4.2). Contrary to the Temporal Module that has N separate Temporal Encoders, the Delta encoders will all share the same trainable weights independently of the temporal window to which X_Δ belongs.

4.4 Training Procedure

4.4.1 Forward Pass

The detection of stealthy attacks against CAN necessitates monitoring large window sizes of CAN messages (signals). However, large input sequences have always been a major problem for various deep learning models resulting in vanishing gradient for LSTMs and quadratic time increase for Transformers. To mitigate this issue, for each time window w, we first feed the signals that are handled by their respective ID to their corresponding temporal encoder, i.e., although if a huge time lapse occurs between same-ID signals, the generated temporal encodings contain the signals' contextual representations. Once all encodings are obtained, they are fed to the delta encoder which iteratively processes and outputs the masked inputs for each interval $\Delta_T << w$.

4.4.2 Backward Pass

Although the iterative process handled by the delta encoder is time-consuming when training, it is advantageous for model optimization during backpropagation. In fact, after each Δ_T, the model's loss is backpropagated and the model's learnable parameters are updated. For instance, if $\Delta_T = 250$ and $w = 5000$, we will perform 20 backpropagations per batch.

To train the AMICA model using self-supervised learning techniques, we leverage the masked language modeling (MLM) objective function, proposed originally for training BERT [43]. The MLM objective consists in masking a percentage of the input sequence at random, and then predicting the masked signals using the output representations. The masked signals will be forecasted by leveraging contextual representation from non-masked input signals, thus the bi-directional capabilities of the model. For the AMICA model, we did not embed the input payload values into a discrete vocabulary. Therefore, the masked signals are replaced by a predefined mask-value that is chosen empirically. In addition, we consider the mean squared error loss function (depicted in Equation 4.3) as the regression-based function for training:

$$L = \sum_{i}^{M} MSE(\widehat{y}_{[Mask_i]}, x_{[Mask_i]}) \tag{4.3}$$

where M is total number of masked signals, $\widehat{y}$ is the prediction of the $i-th$ masked signal, and x is the original value of the $i-th$ masked signal.

4.5 Anomaly Score

The quadratic error between the overall masked inputs and their reconstruction are used to predict whether or not the sequence of CAN IDs with their respective signals is anomalous. The sequence is considered anomalous if the total sum of the masked signals' reconstruction error is above a fixed threshold. It's noteworthy to mention that the threshold is determined based on normal data as we are considering a self-supervised learning problem. We thus consider the following formulated labeling criteria:

$$Y = \begin{cases} 1 \text{ (abnormal)} & \text{if } \sum_{\mathbf{t=1}}^{\mathbf{w}} \mathbf{E}(\mathbf{s_t}) \geq \phi \\ 0 \text{ (normal)} & \text{otherwise} \end{cases}$$

where Y is the CAN sequence's label, $\mathbf{E}$ is the reconstruction error function, $\mathbf{s}_t$ is a corresponding masked signal, w is the considered sequence length, and ϕ is the determined threshold.

4.6 Experiment & Results

To evaluate our proposed method, we used the benchmark synthetic CAN dataset ***SynCAN*** [18]. The dataset is composed of 10 different message IDs, each with different amounts of signals per ID and different noisy time frequencies. The data contains signals that are dependent on one or multiple other signals. The test data is composed of 6 subsets of equal time length: plateau, continuous change, playback, flooding, suppress and normal. We refer readers to [18] for further information.

Parameter	Value
N	1
d_{model}	256
d_{ff}	512
h	8
P_{drop}	0.1
m	0.15
Optimizer	Adam
Adam β_1	0.9
Adam β_2	0.999
Learning rate	0.0001
Batch size	16
Sequence window	5000
Δ_T	250
# Epochs	100
# Workers	32

Table 4.1: AMICA model hyperparameters

For training and evaluation, we use the pyTorch framework [13]. All computations are performed on a Tesla V100 with 32 GB of installed physical memory (RAM). We train the network for 80 iterations with batch size 16. Every element in a batch is a series of 5000 consecutive messages. During a single iteration, a back-propagation is performed every 250 time steps ($\Delta_T = 250$) in order to update the network weights. The model's hyperparameters are depicted in Table 4.1.

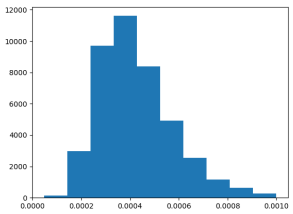

Figure 4.2: Histogram of reconstruction losses for normal CAN data.

Since our approach is self-supervised, we use the normal validation set to compute the

anomaly detection threshold ϕ. As previously discussed, we first calculate the histogram of reconstruction errors for the normal CAN data sequences (seen in Fig. 4.2) and choose empirically ϕ. We thus set ϕ to $\mu + 1.5\sigma = 0.001$ where μ is the mean reconstruction loss on normal data and σ is the standard deviation.

Table 4.2: Performance of AMICA for different attack types

Attack	Recall	Precision	F1-score	AUC
Plateau	0.97	0.81	0.88	0.89
Continuous	0.74	0.85	0.79	0.85
Playback	0.96	0.78	0.86	0.91
Suppress	0.98	0.82	0.89	0.90
Flooding	0.99	0.87	0.93	0.92

We present our results in Table 4.2, Fig. 4.3, and Fig. 4.4. We conclude several performance metrics from the visualized confusion matrices (seen in Fig. 4.3) to assess the performance of AMICA given the threshold ϕ including *precision*, *recall*, *F1-score* (depicted in Table 4.2), in addition to the *ROC* curve along with *AUC* value (shown in Fig. 4.4).

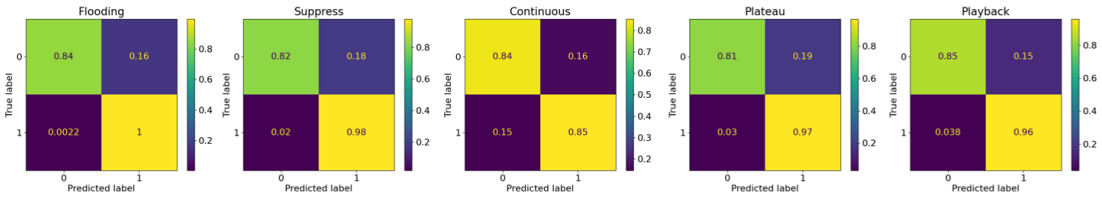

Figure 4.3: Normalized confusion matrices representing the performance of AMICA when detecting diverse attacks on CAN bus for a given threshold ϕ.

Our proposed approach is most prominent when detecting *Timing Transparent (T.T.)* [128] attacks such as *suppress* (F1-score $\approx$ 0.9, AUC $\approx$ 0.9) and *flooding* (F1-score $\approx$ 0.93, AUC $\approx$ 0.92), as these attacks can hypothetically be identified by monitoring the frequency of the CAN IDs, i.e., the appearance of new IDs or disappearance of usually present IDs, irrespective of the values of their carried signals. The *Timing Opaque (T.O.)* [128] attacks including *plateau* (F1-score $\approx$ 0.88, AUC $\approx$ 0.89), *continous* (F1-score $\approx$ 0.79, AUC $\approx$ 0.85), and *playback* (F1-score $\approx$ 0.86, AUC $\approx$ 0.91) attacks are slightly less detectable as they are considered to be stealthier, i.e., they do not disrupt normal timing or ID distributions, and thus would not be detected by simply monitoring the frequency of the CAN IDs. In fact, these attacks solely manipulate the signal values and affect their correlations by overwriting the transmitted signals with constant values (*plateau*), slowly drifted values from their true value (*continuous*), or by playback of

recorded time series of values of that same signal over a period of time (*playback*). Interestingly, our proposed model AMICA excelled in detecting both $T.T.$ and $T.O.$ attacks, whereas the other methods in the literature only fit in frequency-based IDSs or payload-based IDSs.

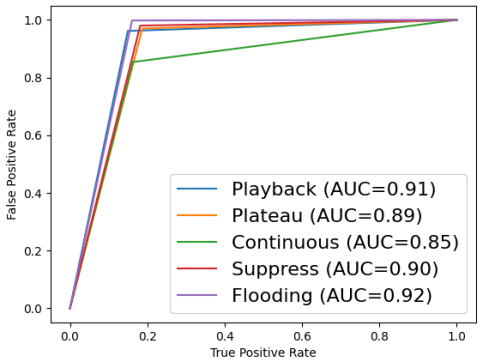

Figure 4.4: ROC curves for diverse types of attack on CAN bus. As the corresponding AUC values approach 1, the AMICA model is thus assumed have a good measure of separability between the normal class and the different attack types.

4.7 Conclusion

We presented a novel deep learning-based multi-agent system, AMICA, for detecting intrusions on the widely deployed in-vehicle communication network CAN bus. As most of the intrusions can only be detected by monitoring long sequences of ordered CAN messages, we develop a model that overcomes this challenge by incorporating information from different relation types between asynchronous signals and IDs, respectively in each stage of the model due to the *attention mechanism*, and by devising a suitable training process. Additionally, unlike the best-known methods so far, our approach is designed to detect all types of attacks on the CAN bus, particularly those that affect the interrelations between asynchronous signals of distinct CAN IDs. To evaluate our method, we leverage the SynCAN dataset and obtain promising results. Our model excels in detecting $T.T.$ attacks, but it is also able to detect stealthier attacks with slightly less performance. The obtained results open many avenues for future research. It would be interesting to compute an anomaly threshold for each signal separately and to study whether this can lead to better model performance, particularly when detecting $T.T.$ attacks. Moreover, it is important to explicitly perform an ablation study on the model architecture along with an exhaustive hyperparameter tuning to enhance its effectiveness and

complexity. Additionally, in both Chapters 3 and 4, we leverage the BERT's model for the detection of attacks on the CAN bus. The devised models have shown remarkable ability in detecting timing opaque and timing transparent attacks. However, the anomaly score for the transmitted network packets offers no information about the cause of the attack. Recently, Li et al. [91] investigated whether language models employ separate mechanisms for different types of linguistic anomalies including morphosyntactic, semantic, and commonsense. They have thus used Gaussian models for density estimation at intermediate layers of three language models (BERT, RoBERTa, and XLNet) and evaluated their method on gathered datasets for anomalies from psycholinguistic studies. They find out that RoBERTa model exhibits surprisal in earlier layers when the anomaly is morphosyntactic than when it is semantic, while commonsense anomalies do not exhibit surprisal at any intermediate layer. Inspired by their work, a current straightforward work objective is to study how our proposed BERT-based models are able to discriminate between attacks that affect the frequency of transmitted IDs and those which affect the value of transmitted signals solely.

Despite CAN's efficiency, it is regarded as a limiting factor for innovative automotive functions in terms of bandwidth, packet size, costs, weight, and higher layer protocols. As the bandwidth requirements in automotive keep increasing, a migration to a cost-efficient high-speed switched network in automotive that lifts these restrictions is necessary. In the next part of this thesis, we study the security of the Automotive Ethernet protocol which is currently adopted by different car manufacturers. to enable the exchange of different kinds of data,i.e., videos, images, graphic data, at a high data rate (up to 1 Gbps) between interrelated electronic control units (ECU). We investigate the vulnerabilities of its application layer protocols SOME/IP and AVTP and propose suitable deep learning-based techniques for intrusion detection and classification.

Part III

Automotive Ethernet

CHAPTER 5

Supervised IDS for the classification of SOME/IP sessions

In this chapter, we present a deep learning-based sequential model for offline intrusion detection on SOME/IP application layer protocol. To assess our intrusion detection system, we have generated and labeled a dataset with several classes representing realistic intrusions, and a normal class - a significant contribution due to the absence of such publicly available datasets. Furthermore, we also propose a recurrent neural network (RNN), as an instance of the deep learning-based sequential model, that we apply to our generated dataset. The numerical results show that RNN excels at predicting in-vehicle intrusions, with F1 Scores and AUC values greater than 0.8 depending on each intrusion type.

Contents

5.1 Introduction

Automobiles are no longer solely made up of mechanical systems. In fact, mechanical components have been taken over by electronics called “Electronic Control Units" ECUs. These connected ECUs through various in-vehicle network infrastructures (CAN, FlexRay, MOST, and LIN) are in charge of making various car functions possible. However, these traditional in-vehicle networks have many limitations in terms of bandwidth and higher-layer protocols. An adaptable and scalable in-vehicle network technology is thus required to realize sophisticated and innovative customer functions such as Adaptive cruise control, Collision avoidance, Driver drowsiness detection, Lane departure warning, and others. To fulfill these automotive requirements, **Automotive Ethernet** technologies have been developed and standardized.

The deployment of Ethernet-based communication in in-vehicle network systems has several other benefits, such as the ability to reuse the associated OSI layers' protocols built and tested in other industries [100]. Furthermore, this cutting-edge technology enables the invention of new protocols for individual layers while reusing protocols for the rest such as the development of the automotive application layer protocol **Scalable service-Oriented Middle-warE over IP** (SOME/IP) [17].

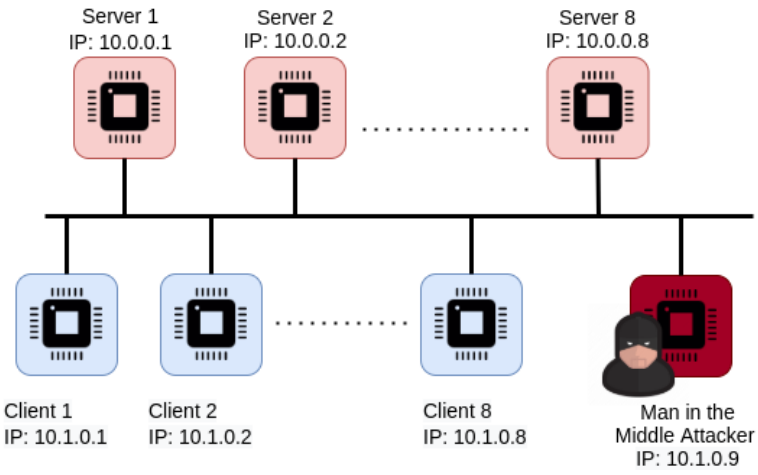

Figure 5.1: Configured Network - Different SOME/IP clients and servers exchanging SOME/IP services over Automotive Ethernet Bus. Besides, a Client ECU is being compromised by a MITM Attacker.

SOME/IP is commonly used for relevant automotive applications due to its service-based communication approach and its adaptability to different automotive operating systems (e.g., QNX, OSEK and Linux) [100]. In other words, SOME/IP is increasingly adopted to coordinate

the exchange of various services between disjoint applications on distinct ECUs. These services cover notifications about in-vehicle events, as well as Remote Procedure Call (RPC) functions that enable an ECU client to request information from an ECU server. However, no security measures, such as authentication or encryption, are defined in the SOME/IP protocol specification [62]. In fact, the absence of SOME/IP security protocols may set the ground for an attacker to exploit a legitimate automotive system and initiate attacks from inside the network, such as intercepting and manipulating messages between two ECUs and other significant threats. To reduce the risk of the various inherent security threats, a robust defense plan is needed, which first requires detecting and analyzing these vulnerabilities.

Due to their large approximation capacity, deep learning-based approaches are well-suited to detect network intrusions in various network types [122] [78]. In this work, we have developed a deep learning-based sequential model to detect network intrusions on the SOME/IP protocol. Sequential models are a category of deep learning model, where the training set is known (a-priori) to have a dominant temporal or causal component: indeed, packets in a session of the SOME/IP protocol exhibit a strong temporal correlation, as each packet depends on previous ones. In the current work, we will contribute to the development of a sequence-based SOME/IP dataset, as no public SOME/IP dataset exists. Specifically, we generate and label a SOME/IP dataset, with four classes of general intrusion packets, as well as a class of normal packets. Moreover, our proposed deep learning-based model, a recurrent neural network (RNN) is able to classify these four intrusions on packets' sequences and the normal ones, with very large accuracy and F1 score. Furthermore, we will evaluate our deep learning-based sequential model using the generated dataset.

Towards this end, this chapter is organized into six sections. Section 5.2 discusses the main publications that are related to SOME/IP intrusion detection. In section 5.3, we present an overview of the SOME/IP protocol. In section 5.4, we present our dataset and the different considered attacks. The suggested sequential model is presented in 5.5. In section 5.6, we present the different evaluation metrics used for performance evaluation. The experimental results discussion is provided in section 5.7.

5.2 Related Work

Deep Learning approaches were highly used in previous works to detect network intrusions on the traditional in-vehicle network protocol **CAN** [49] [71] [120] [78]. However, no previous work

has been addressed to detect intrusions on Automotive Ethernet especially SOME/IP protocol using Deep Learning due to the following reasons.

- **Lack of Automotive Ethernet dataset:** The existence of large CAN databases [54] containing both normal and abnormal network traffic behaviour has resulted in extensive research into deep learning applications on CAN. However, SOME/IP application layer protocol does not have well-known dataset available. Despite the fact that a new Automotive Ethernet dataset is recently published [73], it is not helpful for our current work since it covers normal and abnormal streams of audio-video transport protocol (AVTP) which is different than SOME/IP protocol. Thus, the generation of the labeled dataset (and its publication) is one (but not the only) contribution of the current paper.
- **Automotive Ethernet Standard gaining momentum:** Automotive Ethernet, a recent network protocol for vehicles, is gaining increasing momentum in standards for connected vehicles.

In terms of SOME/IP's latest security vulnerability investigations, researchers have begun to investigate its key vulnerabilities that could lead to cyberattacks on the in-vehicle network, as well as to develop IDS using different approaches. Gehrmann et al. [48] addressed specific problems and opportunities for intrusion detection in SOME/IP, as well as suggested an architecture for a SOME/IP intrusion detection scheme and discussed its security features. Iorio et al. [69] [68] proposed a novel architecture to enhance the security of evolving SOME/IP middleware. Li at al. [93] developed Ori, a Greybox Fuzzer that can efficiently detect breaches in SOME/IP applications. Lauser et al. [84] have discussed how formal models can be used to verify the security of protocols used in modern vehicles. Rumez et al. [116] explained various security countermeasures in the fields of firewalls, intrusion detection systems (IDSs), and identity and access management. Herold et al. [62] proposed a rule-based IDS for SOME/IP protocol.

To the best of our knowledge, none of the previous works have applied deep learning-based sequential models for intrusion detection on SOME/IP protocol. That is a main contribution of this work, in addition to the generation of the labeled dataset (with multiple classes of intrusions and a normal class).

5.3 Overview of SOME/IP

A **"Middleware"** refers to a connective tissue between different software applications. In other words, it handles all functions that are needed for a **"service"** to allow data exchange between

several ECUs[100]. Due to the growing amount of software [34] in automobiles, as well as the spread of functions within their in-vehicle network and the deployment of a variety of software architectures and operating systems inside vehicles, the implementation of a middleware software within in-vehicle networks is essential in bridging the gap between them. Hence, after being proposed by the BMW Group in 2011 and standardized by AUTOSAR, SOME/IP was chosen as the standard middleware for IP-based service-oriented communication in cars. The middleware SOME/IP runs at top levels of the OSI model [48].

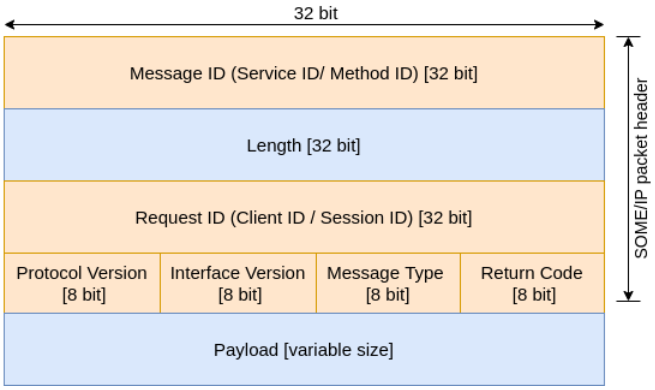

Figure 5.2: SOME/IP packet

The structure of its header layout is as shown in Figure 5.2. Some of the fields presented in the header of the SOME/IP packet will be considered in our work as the input features for the deep learning based IDS (Table 5.4).

5.3.1 SOME/IP Remote Procedure Calls

Since SOME/IP is a service-based communication approach, it allows the exchange of different types of remote procedure calls. In general, a remote procedure call RPC is an inter-process communication technique that is used for client-server based applications [17]. In this current work, we have considered three main types of SOME/IP RPC :

- **Request/Response**: a method with Request and Response messages. The Request is a message sent by the client when it invokes a method. The Response is a message sent from the server to the client that contains the method invocation's outcome.
- **Fire and Forget**: a procedure that only uses Request messages. As in the Request/Response scenario, the client calls a server method. However, unlike in the Request/Response instance, the client does not anticipate a response.

- **Events**: In this method, the server sends messages to the client with particular information either periodically or whenever there is a change (event). The server expects no response from the client.

.

5.4 Generating Labeled Dataset

5.4.1 Dataset Generation

5.4.1.1 SOME/IP packet generator

Class	Training Dataset	Testing Dataset
Normal	2533	2471
Error on Error	39	54
Error on Event	60	54
Missing Response	92	81
Missing Request	83	111
Total	**2807**	**2771**

Table 5.1: Training and Testing Dataset classes

In order to generate SOME/IP libpcap dump files, we have used the SOME/IP Generator developed by [62], implemented in Python 3 and available in Github [16]. The generator models the behavior of different clients and servers assumed to behave according to the AUTOSAR standard specification, as well as an attacker carrying out a variety of attacks depicted in Figure 5.3 and described in section 5.4.1.3. As seen in Figure 5.4 and Table 6.1, we have tuned the different parameters for generating different attack scenarios such as the network architecture configuration depicted in Figure 5.1, the SOME/IP services to be exchanged, and the attack to be implemented along with its frequency of execution. For training and testing our deep learning based IDS, we have generated several pcap files corresponding to different attack types, concatenated them and processed them as described in section 5.4.1.4. The distributions of both datasets are shown in Table 5.1, and their corresponding features are described in Table 5.4. The training dataset comprises about 274 attacks, is 132 MB in size, and contains 2807 packets. Regarding the testing dataset, it contains around 300 attacks, has a size of 130 MB and composed of 2771 packets. Readers can get our SOME/IP intrusion dataset by referring to [14].

Parameters to configure	Description	Chosen Value
Devices	Contains information like name, type, mac, ip sender port and receiving port of each Client, Server and Attacker	8 Servers, 8 Clients and 1 Attacker
Services	Contains information about offered and requested services	3 Services
Number of packets to generate per Client, Method and Service	Defines the number of packets generated per client	50
Number of attacks to execute	Defines the rate an attack will be performed	10
Minimum Response Time of Attacker	Defines the minimum response time of the attacker in ms	1
Maximum Response Time of Attacker	Defines the maximum response time of the attacker in ms	3
Implemented Attack	Defines which attacks can be used	Error On Error Error On Event Missing Request Missing Response
Output File Location	Describes the location where to store the resulting pcap	output.pcap

Table 5.2: Tuned Parameters for Dataset Generation (represents first step in Figure 5.4)

5.4.1.2 Dataset Imbalance

As seen in table 5.1, the distribution of samples across the different classes is biased. In fact, the attack classes frequency is highly imbalanced, i.e., there is a bias or skewness towards the majority class (Normal class) present in the target. It is reasonable to have such a skewed dataset since it represents an anomaly problem. However, we do not aim changing the nature of the data and make it balanced even though this problem poses a challenge for predictive modeling as most of the supervised deep learning algorithms used for classification were designed around the assumption of an equal number of examples for each class. An alternative solution would be the adoption of specialized techniques such as **Adaptive Weighting** [67]. This technique, implemented in our work, is considered as a popular approach for imbalance learning since it weighs samples in rare classes with high cost and then applies cost-sensitive learning methods to deal with imbalance in classes. Table 5.3 presents the weights assigned for each class, for the dataset considered in this work.

Class	Class Weight
Normal	0.16
Error on Event	6.68
Error on Error	10.35
Missing Response	4.33
Missing Request	4.86

Table 5.3: Class Weight

5.4.1.3 SOME/IP intrusions

The SOME/IP attacker is able to compromise a known device within the system. Thus, it has a valid MAC address, IP address, and service ID. It eavesdrops on all traffic within the network

and send packets to all clients and all servers, and thus impersonates other SOME/IP devices and services [62]. Through our work, we are interested in cyberattacks which lead to deviations from the protocol specifications in a communication session between two devices (as seen in Figure 5.3) and which can be detected using deep learning based sequential Models. These four intrusion types considered in this work are detailed below (illustrated also in Figure 5.3):

- **Requests without Response**: Requests have to be answered with either a response or an error message. If a request was never answered, it means that an attacker has relayed the communication between the client and the server who believe that they are directly communicating with each other.

- **Response without Request**: A response should only be delivered in response to an open, previous request. As a result, a normal request with message type 0x00 should be answered by a single response with message type 0x80. Two replies to a single request break the protocol and may indicate the existence of an attacker attempting to impersonate the server and injecting extra packets.

- **Error on Error**: Based on AUTOSAR standard specification, an error message should not be answered with another error message. Hence an incoming error which doesn't have a corresponding request (or other packet) with the same settings indicates the presence of a network intrusion.

- **Error on Event**: Notifications should not be answered with an error message. Thus, a notification replied to with an error depicts a network intrusion between the client and the server.

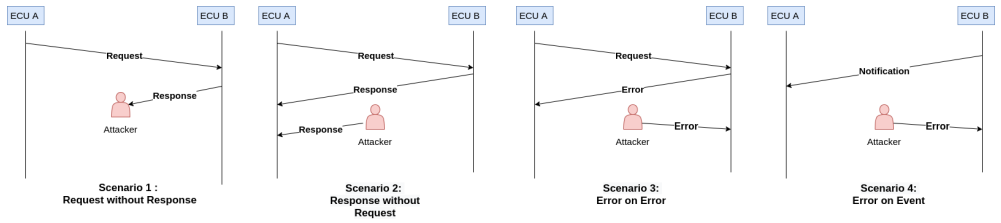

Figure 5.3: Attacks on SOME/IP protocol

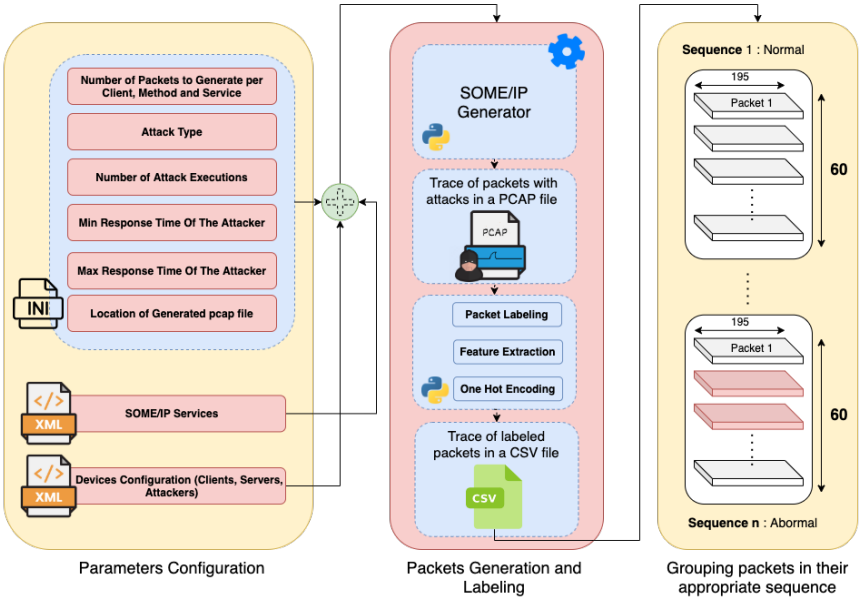

Figure 5.4: Dataset Generation

5.4.1.4 Data Preparation

This section describes the different steps (seen in Figure 5.4) achieved for our dataset to be fed to our deep learning based IDS for training and testing.

1. **Packets generation and labeling:** The SOME/IP packet generator is able to generate pcap files composed of unlabeled packets gathered from the whole network. Since we are using a supervised learning approach, we had to label each packet. Hence, a packet is labeled by 0 if it behaves according to the AUTOSAR standard specification. Otherwise, it is labeled by 1,2,3 or 4 if it represents error on event, error on error, request without response or response without request attacks respectively.

2. **Packets Feature Extraction and One-Hot Encoding:** Each packet is represented by 16 categorical features, described in Table 5.4. However, these features had to be converted to binary vectors using one-hot encoding technique. In fact, many deep learning algorithms cannot work with categorical data directly. Hence, the categories must be converted into numbers. This is required for both input and output variables that are categorical. After encoding, the 15 features that represented input variables were extended to 195 features and the output variable (Label) was extended to 5 classes.

Feature	Description
Service ID	A unique identifier for a service
Method ID	A unique identifier of a method, an event or a field that belong to the service
Client ID	Allows a server to differentiate calls from multiple clients to the same method
Message Type	Used to differentiate different types of messages such as : request,request no return, notification, response and error
Session Id	Allows a subscriber to differentiate multiple calls to the same method
Interface Version	Contains the Major Version of the Service Interface
Protocol Version	Contains the SOME/IP protocol version
Return Code	Used to signal whether a request was successfully processed
IP source	IP of the sending device
IP destination	IP of the receiving device
Protocol	Application layer protocol
Source Port	Port number of the sending device
Destination Port	Port number of the receiving device
Mac source	MAC Address of the sending device
Mac destination	MAC Address of the receiving device
Label	Specifies the class of each packet such as normal, error on error, error on event, request without response, response without request

Table 5.4: Dataset Features

3. **Grouping Packets into Sequences:** In order to detect intrusions affecting the communication behavior between two devices, we had to group packets that belong to each communication in their appropriate sequence. Hence, each sequence represents a series of ordered packets exchanged between a client and a server with the same session identifier. As seen in Figure 5.4, we have grouped packets in their corresponding sequences. Thus, our IDS will detect the presence of an intrusion in a communication between a client and a server by inspecting each sequence of packets. However, since we are dealing with variable length sequence prediction problems, our data had be transformed such that each sequence has the same length. Hence, after transformation, each sequence contains 60 packets which is the maximum number of packets per sequence, i.e, a sequence is padded by zeros if it contains less than 60 packets per session. Our dataset was generated with the constraint that only one type of attack can occur between two devices. Therefore,

sequences are either labeled as normal or by a number corresponding to only one of the (four) possible intrusions. Furthermore, an attack begins and ends in the same session between a client and a server. Hence, an attack cannot be executed in different sessions at the same time.

4. **Sequences Concatenation:** Finally, after labeling the different sequences that represent diverse attacks, we have concatenated them into a single dataset that will be used for training and testing the deep learning-based IDS.

5.5 Proposed Sequential Model

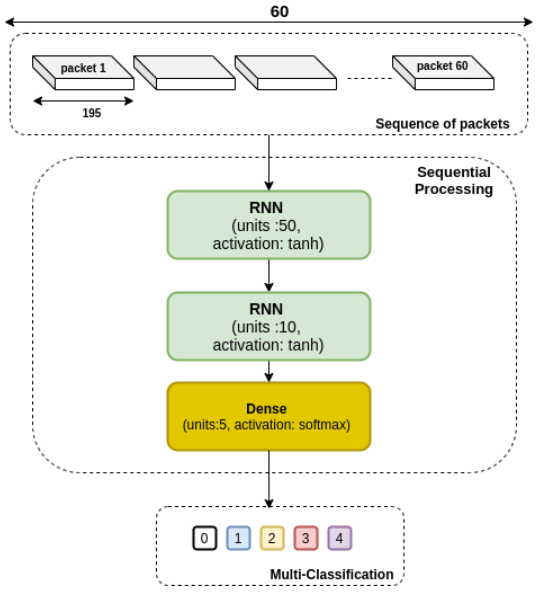

Figure 5.5: RNN-based IDS architecture

Deep Learning based sequential models have been widely adopted to detect intrusions and anomalies in various type of computer networks [139] [113] [66]. Our proposal in this section is to employ Recurrent Neural Networks (RNNs) as a sequential model to the labeled dataset generated in the previous section. The proposed RNN is presented in Figure 5.5. Furthermore, its resulting hyperparameters are shown in Table 5.5. The input to the RNN consists of 60 ordered packets with 195 features each. It passes to two stacked RNN layers which have recurrent connections between hidden units. The two RNN layers read the entire input sequence of 60 packets and feed their output to a dense layer which produces 5 outputs (corresponding to

Hyperparameters	Values
Number of layers	3
Number of Neurons per layer	(50,10,5)
Activation Function per layer	(tanh,tanh,softmax)
Optimizer	Adam
Loss	Categorical Cross Entropy
Learning Rate	0.001
Batch size	100
Epoch size	50

Table 5.5: Hyperparameters

each the 5 classes) using softmax function. We denote the training set, $\{(x_t, y_t)\}_{t=1}^{T}$, where x_t is the feature vector (a vector of dimension 195) for sample $t \in \{1, ..., T\}$, y_t represents the corresponding label for sample $t \in \{1, ..., T\}$, and T the number of samples in the training set. Moreover, each label in the training set is such that, y_t is a binary vector of dimension 5, i.e., $y_t \in \mathbb{B}^5$, where element $i \in \{1, ..., 5\}$ of the vector y_t is a binary variable representing whether the corresponding feature vector, x_t, belongs, i.e., corresp entry = 1 (or does not belong, i.e., corres entry = 0) to class $i \in \{1, ..., 5\}$. Furthermore, y_t may only have one non-zero entry, which follows from our previous assumption that only one intrusion is possible in each sample.

The equations describing the operation for the RNN are the following :

$$a_t = W h_{t-1} + U x_t + b \ , \quad \forall t \in \{1, ..., T\}. \tag{5.1}$$

$$h_t = \psi(a_t) \tag{5.2}$$

$$o_t = V h_t + c \tag{5.3}$$

$$\widehat{y_t} = \phi(o_t) \tag{5.4}$$

where the vector a_t is a linear combination between x_t, the feature vector for sample t, and the hidden layer output of the RNN for sample $t-1$, h_{t-1}. h_t is a vector modeling the hidden layer output of the RNN for a sample t. o_t (a vector of dimension 5) is a linear combination of the output hidden layer h_t. $\widehat{y_t}$ is the prediction that RNN outputs for sample x_t , and has the same properties as y_t. W, U, V, are the shared weights matrices that will be optimized in training. ψ and ϕ are non-linear activation functions, applied element-by-element on their respective inputs.

5.6 Evaluation Metrics

We use the Area Under The Curve (AUC) values, Receiver Operating Characteristics (ROC) curves and F1 scores and calculate them for each class to assess our IDS performance.

We also present the multi-class confusion matrices, which contains information about the actual and prediction classifications done by the classifier, to describe the performance of the multi-classifier models. The training samples corresponding to the label (ground truth) y_t, are represented by each row of the matrix, whereas the occurrences in a predicted label $\widehat{y_t}$ (RNN output), are represented by each column. Specifically, for the task at hand, the confusion matrix will be a 5×5, where element $(i, j) \in \{1, ..., 5\} \times \{1, ..., 5\}$ denotes the normalized number of occurrences , where the true label is from class $i \in \{1, ..., 5\}$, and the predicted label is from class $j \in \{1, ..., 5\}$. Thus, for an ideal multi-class classifier all the diagonal entries should be 1, while the off-diagonal entries should be 0.

In addition to the confusion matrix, we use the F1-score metric explained in Chapter 2.

In the experiments, we use the Python library Keras [12] to implement our RNN model. We train and evaluate our model on an Intel(R) Core(TM) i5-6440HQ CPU @ 2.60GHz.

5.7 Results

Using the generated dataset, we ran a three-fold cross-validation with early stopping to ensure large statistical confidence for our model's prediction performance. In each cross-validation, 67% and 33% of the data are chosen at random as the training and validation sets, respectively. The training set is used for model fitting and the validation set is used for model evaluation for each of the hyperparameter sets. Furthermore, they have the same proportion of classes in each validation fold. After cross-validation, we got three trained RNN models. To assess the overall performance of our approach, we ran three experiments on the testing dataset, one for each trained model.

The classification results for three-fold cross-validation are shown in Table 5.6. The experimental results demonstrated that the model performed well, with acceptable F1-score values for each class of the validation folds. Thus, the models can classify almost all type of attacks on sequences correctly. Moreover, no significant difference in performance metrics exists across the three cross-validations. As a result, the training process is robust with the selected hyperparameters.

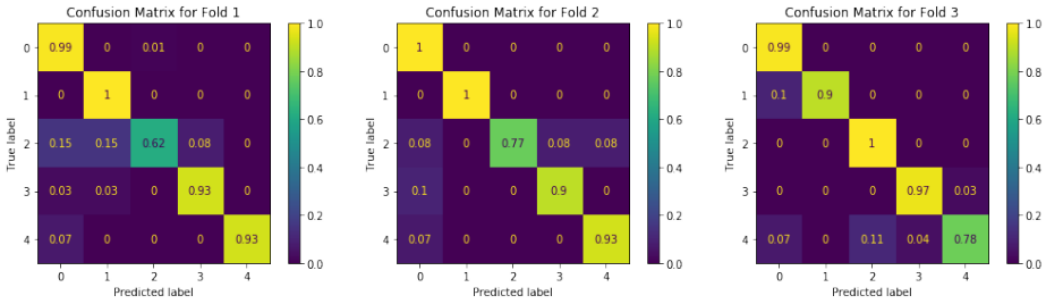

Figure 5.6: Confusion matrices for three different models on Validation dataset

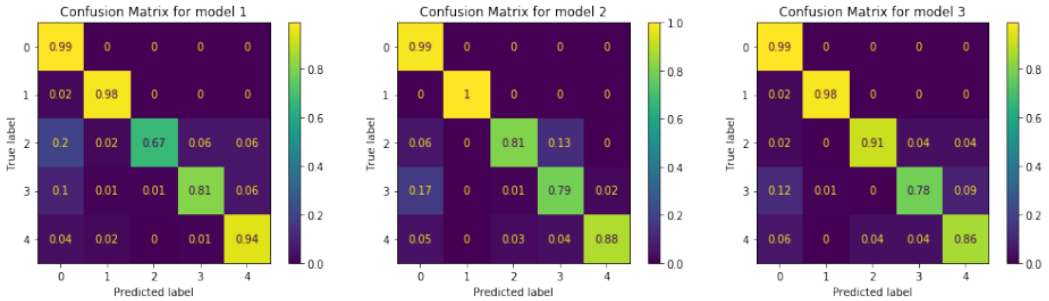

Figure 5.7: Confusion matrices for three different models on Testing dataset

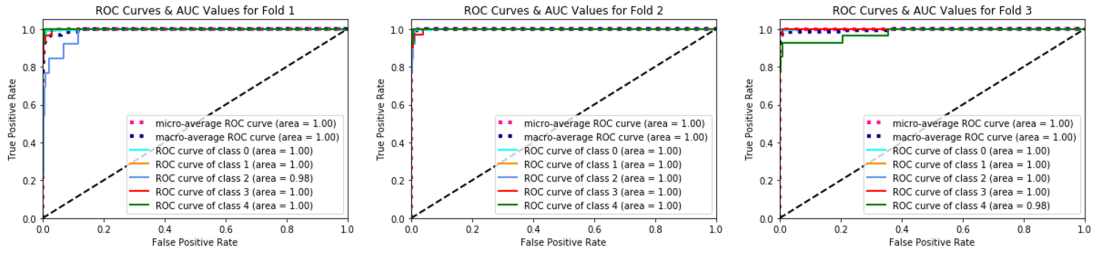

Figure 5.8: ROC Curves and AUC values of each class of the Validation datasets

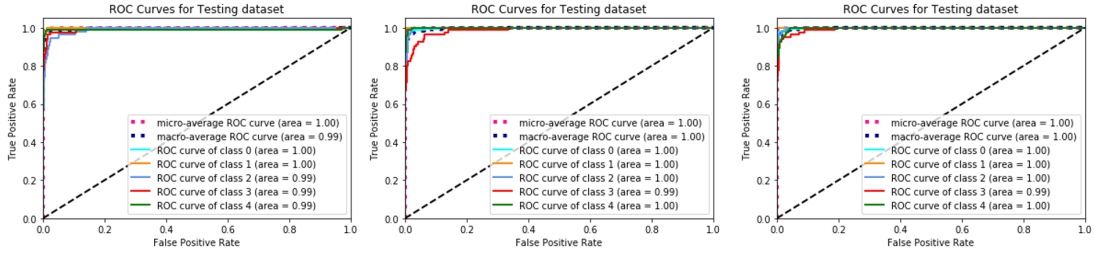

Figure 5.9: ROC Curves and AUC values of each class of the Testing dataset

		Validation			Testing		
Fold	**Class**	**Recall**	**Precision**	**F1-Score**	**Recall**	**Precision**	**F1-Score**
1	Normal	0.99	0.99	0.99	0.99	0.99	0.99
	Error on Event	1	0.87	0.93	0.98	0.93	0.95
	Error on Error	0.61	0.61	0.61	0.67	0.97	0.79
	Missing Response	0.93	0.93	0.93	0.81	0.88	0.84
	Missing Request	0.93	1	0.96	0.94	0.93	0.93
2	Normal	0.99	0.99	0.99	0.99	0.99	0.99
	Error on Event	1	0.95	0.97	1	0.93	0.96
	Error on Error	0.77	0.91	0.83	0.81	0.81	0.81
	Missing Response	0.90	0.93	0.91	0.79	0.79	0.79
	Missing Request	0.93	0.96	0.95	0.88	0.96	0.92
3	Normal	0.99	0.99	0.99	0.99	0.99	0.99
	Error on Event	0.9	1	0.95	0.98	0.98	0.98
	Error on Error	1	0.87	0.93	0.91	0.82	0.86
	Missing Response	0.97	0.88	0.93	0.78	0.84	0.81
	Missing Request	0.77	0.87	0.82	0.86	0.89	0.87

Table 5.6: Results on Validation and Testing Data

Figure 5.6 show a summary of prediction results on the 3 folds for the multi-classification intrusion detection problem during cross-validation. The models have few prediction errors as values outside the diagonal of the confusion matrices approach zero. Hence, the models have well performed since most of the samples are located in the diagonal of the confusion matrices.

Figure 5.8 presents the ROC curves and AUC values for each attack type and for each model (micro-ROC and macro-ROC curves) during the three-fold cross-validation. The displayed figures has AUC values near 1 which means the 3 models have a good measure of separability for the different attack types. Furthermore, the different roc curves have a point in the upper left corner or coordinate (0,1) of the ROC space for each model, representing the ability of the model to have a huge sensitivity (no false negatives) and an outstanding specificity (no false positives). We then performed three tests using the three models that were trained during the cross-validation on the testing dataset. Based on the results shown in Table 5.6, Figure 5.7 and 5.9, we found that the overall performance of the models is outstanding. In fact, the trained models were able to generalize to data that they haven't seen before and did not merely learn to model the training data. On average, the model has well predicted the normal behavior of packets in a sequence (F1-score =0.99). It is also able to predict several other types of attack since the F1-score value varies between 0.8 and 0.96. Moreover, the models' outstanding performance is depicted in Figure 5.9 since the ROC curves of each class are closer to the top-left corner and the AUC values for the different classes approach 1.

5.8 Conclusion

In this chapter, we proposed a RNN-based IDS that detects SOME/IP protocol violations such as communication patterns that are not SOME/IP-compliant by spanning an entire SOME/IP session. It thus classifies the session based on the intrusion type. As our proposed IDS entails two phases: a training phase to learn normal and attack patterns on SOME/IP and an observation and classification phase, we thus generated and labeled a dataset that covers normal and abnormal SOME/IP sessions. Our experiments have shown that the proposed model can be successfully implemented to detect multiple types of intrusions, with very large F1-Scores and AUC values for each class. We believe that the proposed solution for SOME/IP session monitoring is advantageous particularly if the attack patterns include malicious packets that are potentially difficult for a per-packet detection algorithm to distinguish from normal packets. Compared to [81], our solution won't be resource-intensive once deployed on an ECU as the inspected features for each SOME/IP packet in its corresponding SOME/IP service session are directly collected from the packets' headers. However, despite being lightweight, this may potentially lead to a decrease in its ability to detect sophisticated attacks that require monitoring specific features that are in turn resource-intensive once stored for the whole SOME/IP session. Thus, if the ECU that will be equipped with the proposed IDS doesn't have sufficient storage capacity, a trade-off between the selected features and the attack detection robustness will occur. Additionally, our proposed solution was only tested for protocol violation attacks as no tools were at the time of the experiments provided for launching stealthier attacks on SOME/IP. We believe that the devised IDS would potentially be more robust and thus better at detecting sophisticated attacks on SOME/IP sessions if the following features are recorded for each packet [81]:

- The number of concurrent service sessions, that is, the number of concurrent service subscription ECUs.
- The communication interval between packets: such as the communication interval between present Response and previous Request, and present Response and previous Request.
- The difference in payloads between present Response and previous Response, and present Request and previous Request.

In the next chapter, we eschew recurrence in favor of adapting Vaswani's transformer model [127], which permits significantly more parallelization and is, therefore, suitable for real-time

intrusion detection. In contrast to our contributions in this chapter, rather than monitoring the end-to-end session between the SOME/IP clients and the servers, we seek to detect if there is any attacked message on the SOME/IP protocol for a period of time.

CHAPTER 6

SAID: A SOME/IP Attention-based mechanism for Intrusion Detection

In this chapter, we address the challenge of detecting attacks on SOME/IP protocol by monitoring high dimensional in-vehicle network traffic. To overcome the limitations of specification-based and conventional machine learning-based techniques, we present "SAID", a novel attention-based technique for the detection of anomalies from a large sequence of exchanged SOME/IP network packets. For this purpose, we generate two simulated and manually annotated SOME/IP datasets, with different attack ratios, built from the SOME/IP generator tool [1] and which are bigger than the dataset covered in Chapter 5. A comparative study with various deep learning algorithms is performed to show the outstanding performance of our proposed detector in quality while being more parallelizable and requiring significantly less time to detect intrusions.

Contents

[1] https://github.com/Egomania/SOME-IP_Generator

6.1 Introduction

Despite their efficiencies, traditional in-vehicle networking technologies,i.e., CAN, FlexRay, and MOST, are regarded as limiting factors for innovative automotive functions in terms of bandwidth, packet size, costs, weight, and higher layer protocols. As the bandwidth requirements in automotive keep increasing, a migration to a cost-efficient high-speed switched network in automotive that lifts these restrictions is necessary. Established by the OPEN Alliance, Automotive Ethernet - or more correctly "Ethernet-based communications" - was since then adopted by different car manufacturers. It enables the exchange of different kinds of data,i.e., videos, images, graphic data, at a high data rate (up to 1 Gbps) between interrelated electronic control units (ECU), thus allowing automated and autonomous driving functionalities (advanced driving assistance system (ADAS), adaptive cruise control), infotainment services as well as speedy diagnosis and flash updates. Additionally, another driving force behind the adoption of this technology is that it allows the development of new automotive protocols for specific layers within the ISO/OSI models while allowing the reuse of protocols for the remaining others. However, the increasing automotive software complexity necessitates switching to service-based in-vehicle communication. The Scalable service-Oriented MiddlewarE over IP (SOME/IP) is an automotive middleware protocol that operates at the higher layers of the ISO/OSI layer model. Its key advantages lie in the complexity reduction of the Ethernet-based in-vehicle network by providing serialization, remote procedure call (RPC), and service discovery, among other features. In fact, it organizes and controls data exchange over the in-vehicle network between decoupled software components spread over various processes in different ECUs which support different operating systems (POSIX, QNX), thereby enabling distributed functions development.

Unfortunately, as in-vehicle software becomes more sophisticated and vehicles get increasingly connected, automobiles will inevitably encounter stealthy cyberattacks that can have severe consequences on personal safety and privacy. More importantly, the attack can be spread to all vehicles of the same type. Researchers [47] have recently found relevant vulnerabilities in SOME/IP protocol which in turn can lead to exploitation and car hacking,i.e., man-in-the-middle (MITM) attack [143] in which an attacker can intercept, manipulate, and interrupt the communication between different ECUs. To mitigate these risks, the international regulation UN R-155 mandates cybersecurity monitoring to assess vehicle data and records for vulnerabilities and cyberattacks. Consequently, a wide array of security controls, (e.g. authentication, encryption, firewalls, and secure updates) have been proposed to guarantee the security of SOME/IP protocol,

and mitigate, and prohibit targeted attacks from growing substantially [69] [68][93] [84] [116]. In this paper, we consider developing a SOME/IP-specific Intrusion Detection System (IDS) that is able to monitor suspicious SOME/IP network traffic behavior and detect potential intrusions.

While the proposed state-of-the-art solutions leverage specification-based techniques [48][62] [82] which in turn require substantial human effort for crafting suitable specification rules or features, our research overcomes this key challenge by leveraging suitable deep learning techniques that extract high-level, abstract features, from raw sequences of exchanged SOME/IP packets and which in turn facilitate the real-time intrusion detection task. More importantly, given the problems of the large volume of SOME/IP network traffic due to the emergent dynamical structure of the in-vehicle network, the deep learning approaches are regarded as a good candidate for intrusion detection owing to their ability to process high-dimensional data.

In contrast to our previous work [14], which employs a Recurrent Neural Network (RNN) for offline classification attacks on SOME/IP protocol, we eschew recurrence in favor of adapting Vaswani's transformer model [127], which permits significantly more parallelization and is, therefore, suitable for the real-time intrusion detection. Although we use the transformer's model "self-attention" mechanism to enable modeling of the interdependencies between different elements of the input network sequence regardless of their distance, our proposed neural network architecture is considerably simpler as it is not built on the typical Encoder-Decoder architecture format for language translation.

Hence, the contribution of this work is three-fold. For one, we introduce "SAID", a neural network-based approach that leverages attention-based mechanisms for identifying anomalies in long sequences of exchanged network packets. Moreover, we simulate and manually annotate two SOME/IP datasets, with different attack ratios, to evaluate our proposed detector. Finally, we compare "SAID" with other deep learning algorithms to validate its outstanding performance, particularly when the attack ratio increases.

This chapter is organized as follows. Section 6.2 discusses main publications that are related to SOME/IP security countermeasures. In Section 6.3, we present the leveraged threat model and the different considered attacks. The generated and labeled dataset is presented in Section 6.4. In Section 6.5, we present our proposed attention-based detector "SAID". We discuss our experimental results in Section 6.6, followed by a conclusion that summarizes our study.

6.2 Related Work

Recent studies have focused on the security investigation of the SOME/IP protocol and the corresponding security countermeasures. Iorio et al. [68] [69] presented a novel mechanism to provide improved security to the emerging SOME/IP middleware, without introducing at the same time limitations in the communication patterns available. Zelle et al. [143] present a formal and practical security analysis of SOME/IP, the identified Man-in-the-Middle (MITM) attacks, and propose two security extensions for authentication and authorization of service provisioning and usage protect against these attacks. Ma et al. [99] designed an efficient secure scheme, including an authentication scheme using the SOME/IP protocol and a secure communication scheme modifying the payload field of the original SOME/IP data frame. Du et al. [47] uses the model building method based on the Colored Petri Net (CPN) theory to model the SOME/IP protocol of the vehicle Ethernet. The security protocol is formally verified and analyzed by combining it with the Dolev–Yao adversary model detection method. After verification, the protocol is subject to three attack vulnerabilities: replay, tampering, and deception. They introduced timestamps and random numbers to strengthen the protocol security. After the final analysis and verification, their improved scheme can effectively improve the security performance of the protocol. Li et al. [93] propose Ori — a greybox fuzzer for SOME/IP applications, which features two key innovations: the attach fuzzing mode and structural mutation. Their evaluation shows that Ori can detect vulnerabilities in SOME/IP applications effectively and efficiently. Koyama et al. [82] proposed an IDS that achieves high accuracy by combining two whitelist-based anomaly detection algorithms: a real-time algorithm that compares received SOME/IP packets with a normal communication model to determine whether they are anomalies and a retroactive algorithm which performs retroactive detection by determining whether packets are anomalous from long-term time characteristics spanning an entire SOME/IP session. Alkhatib et al. [14] presented a deep learning-based sequential model for offline intrusion detection on SOME/IP application layer protocol. Tobias et al. [48] propose an architecture for a SOME/IP intrusion detection system, discuss its security properties and report preliminary experimental results. Heo et al. [61] introduced a new type of intrusion detection system (IDS) leveraging on SOME/IP packet's header information and packet reception time to deal with SOME/IP-related network attacks. Casparsen et al. [36] proposed a host-based IDS for SOME/IP that detects where the detection is based on arrival time, payload values, and packet contradictions. Herold

et al. [62] presented an anomaly detection system using the Esper complex event processing engine, thus applying a domain-specific rule set to a stream of SOME/IP packets.

Numerous researchers are leveraging attention-based models for intrusion detection. Wu et al. [136] proposed RTIDS, a robust transformer-based approach for Intrusion Detection System, which leverages self-attention mechanism to facilitate network traffic type classifications. Tan et al. [123] present a new technique based on the neural attention mechanism for real-time attack detection since it uses time slot-based features. Wang et al. [131] designed a hybrid neural network DDosTC structure, combining efficient and scalable transformers and a convolutional neural network (CNN) to detect distributed denial-of-service (DDoS) attacks on Software Defined Networks (SDN). Yang et al. [138] proposed an intrusion detection model based on an improved vision transformer (ViT) and evaluated using the NSL-KDD dataset to show its outstanding performance over existing intrusion detection models. Ho et al. [63] have also proposed a new intrusion detection method that uses image conversion from network data flow to produce an RGB image that can be classified using the Vision Transfomer (ViT) model [46]. Nam et al. [105] proposed an intrusion detection model that combines two GPT networks in a bi-directional manner which is trained to minimize the negative log-likelihood (NLL) value for a normal sequence. When the NLL value for a CAN ID sequence is larger than a prespecified threshold, it is deemed an intrusion. Alkhatib et al. [27] proposed “CAN-BERT", a BERT-based network intrusion detection system that can learn the sequence of arbitration identifiers (IDs) in the CAN bus for anomaly detection using the “masked language model" unsupervised training objective.

6.3 Threat model

While SOME/IP has useful features, it is lacking essential security properties like authentication and encryption. As a result, we’ve used a threat model previously implemented by [62] in which an adversary behaves as a man-in-the-middle attacker and does not follow the SOME/IP protocol definition. Consequently, we explored a SOME/IP-based network without service discovery characteristics in our study. Clients are aware of all information about the servers, including their MAC and IP addresses. Each client is only authorized to use a restricted number of services and approaches. Some servers transmit notifications to certain clients on a regular basis. The precise intervals utilized for these services, as well as the services and methods associated with them, are known ahead of time. As [62] specified, an attacker can do the following:

- Compromises a known device within the system. Thus, the attacker has a legitimate MAC address, IP address, and service ID.
- Sniffs SOME/IP network traffic exchanged in the in-vehicle network
- Transmits packets to all network participants,i.e., clients and servers, and thereby impersonates other SOME/IP devices and services.

6.3.1 Attacks

We have considered four intrusion types in this work, detailed below :

- **Requests without Response**: Requests have to be answered with either a response or an error message. If a request was never answered, it means that an attacker has relayed the communication between the client and the server who believe that they are directly communicating with each other.
- **Response without Request**: A response should only be delivered in response to an open, previous request. As a result, a normal request with message type 0x00 should be answered by a single response with message type 0x80. Two replies to a single request break the protocol and may indicate the existence of an attacker attempting to impersonate the server and inject extra packets.
- **Error on Error**: Based on AUTOSAR standard specification, an error message should not be answered with another error message. Hence an incoming error that doesn't have a corresponding request (or another packet) with the same settings indicates the presence of a network intrusion.
- **Error on Event**: Notifications should not be answered with an error message. Thus, a notification replied to with an error depicts a network intrusion between the client and the server.

6.4 Dataset Generation

Given a sequence of SOME/IP packets, we aim to detect whether this sequence is normal or anomalous, i.e., a SOME/IP sequence is anomalous if it contains at least one abnormal (i.e., injected/out of order/replayed) packet. Hence, we have generated and labeled a dataset that

Table 6.1: Tuned Parameters for Datasets Generation

Parameters	**Network Configuration 1**	**Network Configuration 2**
Devices	8 Servers 8 Clients 1 Attacker	8 Servers 8 Clients 1 Attacker
Services	3 Services	3 Services
Number of packets to generate per Client and method/service	500	500
Minimum Response Time of Attacker	1	1
Maximum Response Time of Attacker	3	3
Implemented Attack (Attack Ratio)	Error On Error(4.63%) Error On Event (4.41%) Missing Request (8.12 %) Missing Response (6.60 %)	Error On Error (9.95%) Error On Event (9.39%) Missing Request (24.8%) Missing Response (18.28 %)

contains benign and malicious SOME/IP packets using the SOME/IP Generator developed by [62], implemented in Python 3 and available in Github [16]. The generator models the behavior of different clients and servers assumed to behave according to the AUTOSAR standard specification, as well as an attacker carrying out a variety of protocol violation attacks described in Section 6.3.

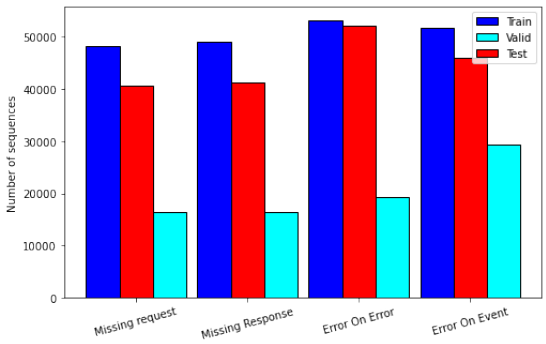

Figure 6.1: Low attack ratio dataset

The datasets are recorded in the PCAP file format and, therefore, are viewed using prevalent programming libraries and packet analyzers (such as Wireshark). As seen in Table 6.1, we have tuned the different parameters for generating different attack scenarios, the SOME/IP services to be exchanged, and the attacks to be implemented along with their frequency of execution. For training and testing our deep learning-based IDS, we have generated several PCAP files corresponding to different attack types with the attack ratio depicted in Table 6.1. The ratio represents the percentage of attacks for a single PCAP file. We have balanced the dataset for

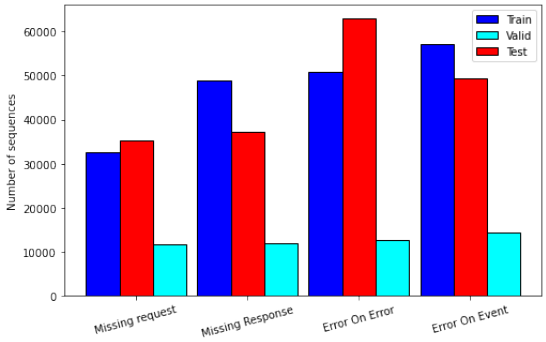

Figure 6.2: High attack ratio dataset

training to avoid any biased learning by the deep learning models. In fact, we considered the same amount of normal and abnormal SOME/IP sequences for training the different approaches. When training, the abnormal set contains different attack types. During testing, we evaluated the performance of SAID for each attack separately. The dataset statistics are represented in Figures 6.1 and 6.2.

Since the IDS will be monitoring the SOME/IP network traffic each 100ms, we collect sequences composed of 128 SOME/IP packets using the **Feature-based Sliding Window (FSW)**, where T represents the window size or the total number of packets per window and the slide size is 1. Hence, each sequence of ordered packets is defined as $S = \{\mathbf{p}_1, ..., \mathbf{p}_t, ..., \mathbf{p}_T\}$, where $\mathbf{p}_t$ indicates a transmitted SOME/IP packet at time t and $\mathbf{y}_t \in \{0, 1\}$ its corresponding label, with 1 indicating a malicious packet and 0 otherwise. Each packet $\mathbf{p}_t$ in the SOME/IP network traffic is represented by 58 features, each of which has an integer value between 0 and 255. Moreover, we label each SOME/IP sequence using the following criteria:

$$y = \begin{cases} 0 \text{ (normal)} & \text{if } \mathbf{y}_t = 0, \forall t \in \{1, .., w\} \\ 1 \text{ (abnormal)} & \text{otherwise} \end{cases}$$

$y \in \{0, 1\}$ is the SOME/IP sequence's label, which is labeled as anomalous if there is at least one anomalous packet.

6.5 Proposed framework: SAID

We now thoroughly explain the architecture of the SOME/IP Attention-based IDS (SAID) which is inspired by the transformer's ability to handle ordered sequences of data [127]. For an overview please refer to Figure 6.3. In contrast to Vaswani's model [127] which leverages

an encoder-decoder structure for machine translation, our proposed detector employs only the encoder network followed by a sigmoid activation layer for binary classification of SOME/IP packets' sequences.

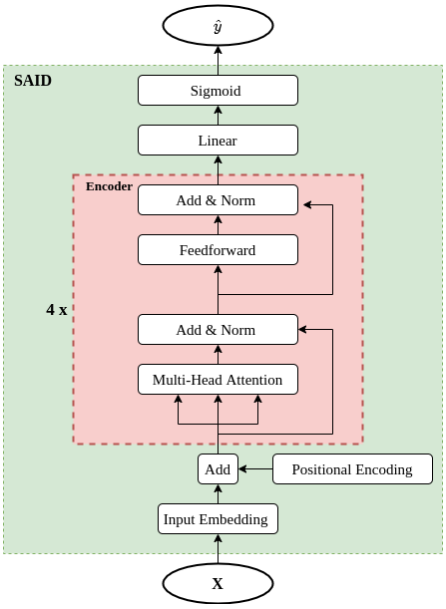

Figure 6.3: The overall structure of our proposed SOME/IP intrusion detector "SAID". We first project every SOME/IP packet into an embedding space and then inject positional encodings and apply dropout. Next, the embeddings are fed to L = 4 stacked attention layers. Finally, we detect intrusions by feeding the resulting output into the sigmoid activation function.

The SOME/IP packets' sequence $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_T\} \in \mathbb{R}^{T \times d}$, is fed firstly into the SAID's "Input Embedding" module. $\mathbf{x}_t$ denotes a d-dimensional SOME/IP packet transmitted in the Ethernet-based in-vehicle network at time t. The Input embedding projects $\mathbf{X}$ into a h-dimensional space using a single linear linear $\mathbf{X}_I = \mathbf{X} \cdot \mathbf{W}_0 + \mathbf{b}_0$. Note that, the weights are represented as $\mathbf{W}_0 \in \mathbb{R}^{d \times h}$, $\mathbf{b}_0$ is the bias, h is the hidden size. $\mathbf{W}$ is randomly initialized and updated with the other parameters of the model during the training phase.

We additionally inject a notion of ordering by adding sinusoidal positional encoding to the SOME/IP packets' embeddings following [127]. The tth SOME/IP packet's position embedding can be represented by the following equations:

$$\mathbb{PE}_{t,2i} = sin(t/10000^{2i/d}), \tag{6.1}$$

$$\mathbb{PE}_{t,2i+1} = cos(t/10000^{2i/d}), \tag{6.2}$$

where i is in the range of $[0, d/2]$, and d is in the input dimension.

The resulting embedding $\mathbf{X}_E = \mathbf{X}_I + \mathbf{X}_P$ is thus fed into the next module with $\mathbf{X}_P \in \mathbb{R}^{T \times h}$ is the positional embeddings of the SOME/IP packets' sequence.

The output $\mathbf{X}_E \in \mathbb{R}^{T \times h}$ obtained from the previous module is passed to a stack of L attention blocks where we apply the "self-attention" attention to update the embeddings. Similarly to [123, 127], we use the scaled dot-product attention, requiring a matrix of keys $\mathbf{K} \in \mathbb{R}^{T \times h}$, a matrix of values $\mathbf{Q} \in \mathbb{R}^{T \times h}$, and a matrix of "queries" $\mathbf{Q} \in \mathbb{R}^{T \times h}$. The attention operation yields a weighted sum of values $\mathbf{V}$:

$$Attn(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \tau(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{h}}) \cdot \mathbf{V} = \mathbf{AV} \tag{6.3}$$

where τ is the softmax function. The resulting output context contains information about the intrinsic dependencies between latent representations of a sequence of SOME/IP network packets $\mathbf{X}$ within a data sample. It will subsequently be used to assist in the predictions of $\widehat{\mathbf{y}}$. Additionally, a residual connection is applied around the attention mechanism, followed by a layer normalization:

$$\mathbf{X}_A = Norm(\mathbf{Q} + \mathbf{AV}) \tag{6.4}$$

After the layer normalization, we pass the output to a fully connected Feed Forward (FF) network and a subsequent layer normalization with the residual connection. Following [123**?**], the feed-forward block consists of two linear projections separated by a ReLU activation:

$$FFN(\mathbf{X}_A) = ReLU(\mathbf{X}_A \cdot \mathbf{W}_1 + \mathbf{b}_1) \cdot \mathbf{W}_2 + \mathbf{b}_2 \tag{6.5}$$

$$\mathbf{X}_{FF} = Norm(\mathbf{X}_A + FFN(\mathbf{X}_A)) \tag{6.6}$$

The variables $\mathbf{W}_1 \in \mathbb{R}^{h \times p}$, $\mathbf{W}_2 \in \mathbb{R}^{p \times h}$, $\mathbf{b}_1 \in \mathbb{R}^p$, and $\mathbf{b}_2 \in \mathbb{R}^h$ are shared parameters across inputs from various sequential positions. The output of Equation 6.6 is finally transmitted to the classification(output) layer.

The output $\mathbf{X}_{FF}$ given by the Encoder module passes through a final fully connected layer with the sigmoid activation function which outputs the binary class $\widehat{\mathbf{y}} \in \{0, 1\}$ of the SOME/IP sequence, with the value 1 representing the possibility of intrusion in a sequence of SOME/IP packets and 0 otherwise.

6.6 Experimental Results

In this experiment, we evaluate the attack detection capabilities of SAID on the high attack ratio and low attack ratio datasets and perform a comparative analysis with other deep learning

techniques including convolutional neural network (CNN), recurrent neural network (RNN), long short-term memory network (LSTM), and convolutional LSTM network (CNN-LSTM). The models have been implemented using Python programming language and Pytorch framework on the Tesla V100S-PCIE-32GB machine. For each dataset, we train the algorithms using the balanced training data and the depicted parameters in Table 7.2. Additionally, we use early stopping to avoid overfitting. We set the window size as 128 throughout our experiment, which considers the temporal and contextual information between SOME/IP packets, memory, and computation efficiency. The deep learning models will fail if the window size is smaller for intrusion detection. Note that, although a larger window size makes the model learn complex contextual interdependencies, it results in larger memory costs for the embedded systems.

Table 6.2: DL Models Parameters

Parameter	**Value**
Learning Rate	0.0001
Optimizer	Adam
Adam β_1	0.9
Adam β_2	0.999
Batch Size	32
Early stopping	Yes
Loss	Binary Cross Entropy

After hyperparameter tuning, we consider the depicted parameters in Table 6.3 for SAID. We consider the following configurations for other deep learning models:

- The **RNN (LSTM)** model used here consists of 2 layers, with the first layer's hidden size set at 16 and the second layer's hidden size set at 8. After each RNN (LSTM) layer, an activation function based on the hyperbolic tangent is applied.

- For the **CNN** model, we convert each sequence into 128×58 size of images. The input is thereby fed to the CNN model composed of 2 convolutional layers, having each a number of channels of 8 and 4 respectively. The kernel size is set to 3 and the stride size is set to 2. Additionally, we choose ReLU as an activation function for each convolutional layer.

- For the **CNN-LSTM** model, a sequence of packets converted to an image of size (128,58) is fed to the CNN model composed of three convolutional layers with 3, 6, and 3 as their number of channels. Their kernel size is set to 3 and their stride size is 1. After that, the

output of the CNN is fed to an LSTM neural network composed of three layers with 16, 8, and 4 as their number of features in the hidden states. We also use ReLU as an activation function for each convolutional and LSTM layer. Finally, a "sigmoid" activation function is used to classify network traffic based on the output of the last layer.

Table 6.3: SAID model configuration

Parameter	Value
L	4
$d_{model}(h)$	8
d_{ff} (p)	32
$\#$ *heads*	1
P_{drop}	0.1
Optimizer	Adam
Adam β_1	0.9
Adam β_2	0.999
Learning rate	0.001
Batch size	32
# Epochs	200
Patience	10

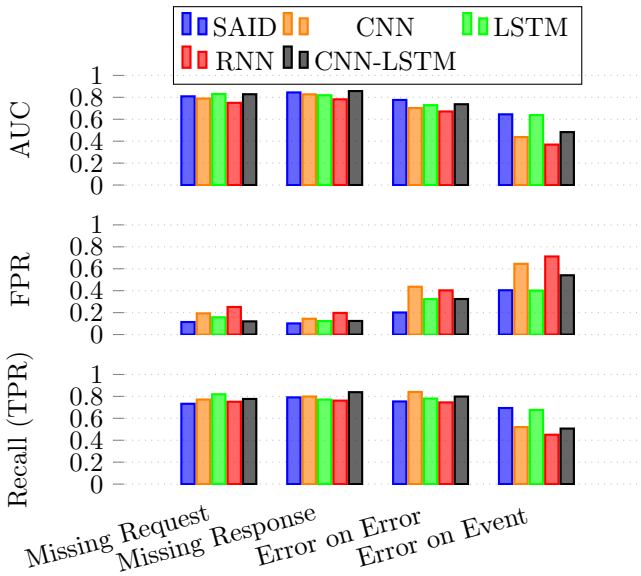

Figure 6.4: Low attack ratio dataset

Figures 7.3 and 7.4 provide the TPR, FPR, and AUC scores for SAID and other deep learning models for both low and high-attack ratio datasets. For the low-attack ratio dataset (seen in 7.3), all models have comparable performance and perform relatively well on the “Missing Request"

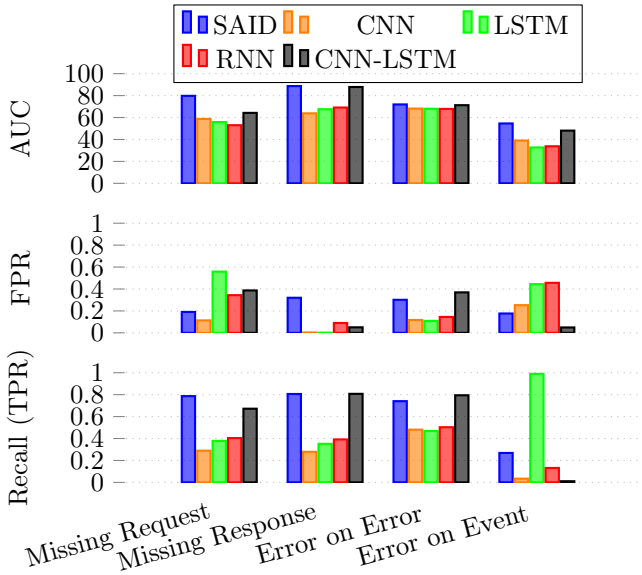

Figure 6.5: High attack ratio dataset

Table 6.4: Models Complexity

Model	Time (ms)	Size (MB)	# Parameters
SAID	0.30 ± 0.30	0.12	1,329
CNN	0.07 ± 0.05	0.16	1,985
LSTM	4.583 ± 0.60	0.08	11,401
RNN	4.786 ± 0.13	0.06	2,857
CNN-LSTM	0.37 ± 0.34	15.53	3,750,896

($\overline{AUC} \approx 0.80$), “Missing Response" ($\overline{AUC} \approx 0.83$), and “Error on Error" ($\overline{AUC} \approx 0.72$) in terms of average AUC score. However, their performance drop significantly when identifying the “Error on Event" intrusion ($\overline{AUC} \approx 51.6$) due to the fact that its behavioral pattern is analogous to normal SOME/IP network traffic, which increases the difficulty for our model and other techniques to correctly identify it. Nevertheless, SAID performs slightly better than the average for all attacks on SOME/IP that we consider in this work (Missing Request AUC $\approx$ 0.81, Missing Response AUC $\approx$ 0.84, Error on Error AUC $\approx$ 0.78, Error on Event AUC $\approx$ 0.65). It means that our model performs well in the false-positive and true-positive rates under various pre-selected thresholds. For the high-attack ratio dataset (seen in Figure 7.4), SAID (Missing Request AUC $\approx$ 0.80, Missing Response AUC $\approx$ 0.89, Error on Error AUC $\approx$ 0.72) outperforms other deep learning algorithms for all attacks in terms of AUC scores (Missing Request $\overline{AUC} \approx$

0.62, Missing Response $\overline{AUC} \approx 0.75$, Error on Error $\overline{AUC} \approx 0.69$). Our proposed detector goes beyond the point-wise representation learned by the conventional deep learning techniques and models the more informative interdependencies between SOME/IP packets. However, although the “Error on Event" attack ratio has increased, all models also perform poorly when detecting this protocol violation ($\overline{AUC} \approx 0.42$). A general comparison of the results from both datasets implies that SAID not only improves the AUC and recall rate of SOME/IP intrusion detection but also decreases the false positive rate for the majority of intrusions.

Eventually, as our solution will be deployed on Electronic Control Units (ECUs) which have tight resource constraints including computing, memory, and important power budget, we assessed their computational performance. The results are listed in Table 6.4. As depicted, while the LSTM and RNN-based models take longer time to detect attacks on SOME/IP protocol, the CNN, CNN-LSTM, and SAID detect intrusions between 0.07 and 0.4 ms. Using SAID allows much faster detection than recurrent methods by parallelizing inference on GPUs. Despite CNN's suitability for real-time detection in terms of inference time, it poorly detects attacks when compared to both CNN-LSTM and SAID. Hence, the experiments testify to the advantage of the proposed framework and verify its effectiveness against the majority of intrusions. It provides the benefit of being able to encode large sequences with accuracy and small inference times.

6.7 Conclusion

As the in-vehicle network is increasingly scaling due to the emerging SOME/IP protocol, the specification-based techniques and conventional machine learning techniques can no longer be adopted as they require substantial human effort for crafting suitable specification rules or features, respectively. Additionally, due to scaling, the high-dimensional nature of transmitted interdependent SOME/IP network packets is particularly challenging for intrusion detection, as many corresponding dimensions may be noisy and irrelevant for anomaly detection. Our research aims to overcome these challenges by the deployment of effective security solutions that adapt to the recent dynamic in-vehicle environment. More concretely, we present a novel deep learning-based IDS for SOME/IP able to learn features with different levels of abstraction at different processing layers without human intervention. Additionally, using the self-attention mechanism, our approach can model the contextual dependencies between SOME/IP packets and which are crucial for the detection of contextual intrusions. The tests performed on SOME/IP

datasets show that our proposed model is computationally efficient and achieves an overall superior performance than the prevailing sequential models, particularly when the attack ratio increases. For future work, we aim to develop novel strategies to further amplify the difference between normal and abnormal SOME/IP network traffic and to investigate the effectiveness of the attention mechanism in improving our model's interpretability. Moreover, as mentioned earlier, SOME/IP can be roughly divided into three parts: Service Discovery (SD), Remote Procedure Call (RPC), and access to process data. Throughout this chapter and the previous one (Chapter 5), we only considered attacks on SOME/IP due to RPC's vulnerabilities. However, the SD feature must be addressed primarily - before the subsequent RPC - since its security will lead to restricted service provisioning and usage, thus, limiting the damage an attacker can cause by compromising an ECU [144]. In fact, the SD feature allows entities to dynamically find services and determine their corresponding IP addresses and port numbers on which these services can be accessed. Using the Publish/Subscribe mechanism of SOME/IP, the clients can then subscribe to services to obtain the data published by these services whenever necessary. In SOME/IP's parlance, a server offers a service instance that a client can subscribe to. Consequently, SOME/IP security mainly revolves around the question of who is allowed to offer which service and who is allowed to subscribe to which services. Our contributions regarding SOME/IP protocol could be improved by the development of a deep learning-based IDS that is not only limited to its RPC feature (as in our case) but rather takes into consideration the SD and Publish/Subscribe features to monitor which services are being offered and subscribed to.

In the next chapter, we seek to detect cyberattacks that cause the interruption media streams which are carried by the Ethernet-based protocol IEEE 1722. In contrast to [74] which aim to detect replay packets using supervised learning techniques, we seek to detect whether a sequence of AVTP packets contains one or several injected packets. Notably, keeping this goal in mind, our proposed study is more general and thus covers models which can be further used for the detection of stealthier cyberattacks. We thereby provide a comparative analysis between different unsupervised deep learning and machine learning methods for the detection of audio-video transport protocol (AVTP) stream injection attacks in automotive Ethernet-based networks.

CHAPTER 7

Real-time unsupervised intrusion detection on AVTP

In this chapter, we compare the performance of different unsupervised deep and machine learning-based anomaly detection algorithms, for real-time detection of anomalies on the Audio Video Transport Protocol (AVTP), an application layer protocol implemented in the recent Automotive Ethernet-based in-vehicle network. The numerical results, conducted on the recently published **"Automotive Ethernet Intrusion Dataset"**, show that deep learning models significantly outperform other state-of-the-art traditional anomaly detection models in machine learning under different experimental settings.

Contents

7.1 Introduction

Since the advent of power electronic components such as sensors and actuators as well as a robust in-vehicle infrastructure for efficient data exchange between them, driving has become safer (i.e. 360- degree surround view parking assistance and collision avoidance systems) [97] and more pleasant (i.e. infotainment features)[109] [124] during the last several decades. Ethernet, a flexible and scalable networking technology in communication systems, is recently standardized and adopted for in-vehicle communication [58][31] between different Electronic Component Units (ECU). In fact, it fulfills basic automotive requirements which existing in-vehicle protocols LIN, CAN, and FlexRay are not designed to cover, including reduced connectivity costs, cabling weight, and support for high data bandwidth.

To ensure low-latency and high-quality transmission of time-critical and prioritized streaming data for high-end infotainment and ADAS systems, the IEEE 1722 audio-video transport protocol (AVTP)[4] is adopted. In fact, AVTP specifies a protocol for audio, video, and control data transportation on a Time-Sensitive Networking (TSN) capable network [102]. As a result, we believe that AVTP protocol will be a critical protocol for Automotive Ethernet-based in-vehicle network in motor vehicles.

Despite the advantages of Automotive Ethernet, the drive toward connectivity has significantly expanded the attack surfaces of automobiles, making Automotive Ethernet-based in-vehicle networks increasingly susceptible to cyberattacks, posing significant security and safety issues [79]. In fact, Automotive Ethernet can be attacked by exploiting its vulnerabilities [15][130]. These security breaches can affect protocols working on top of it, including AVTP protocol, and might therefore lead to the interruption of critical media streams.

To address this, intrusion detection systems (IDS) should be used in addition to specific security measures as an extra layer of protection. These systems can be classified based on their analyzed activity (i.e., monitoring a network or a host activity logs) and their detection approach (i.e., signature-based or anomaly-based detection). Deep learning models, usually referred to as anomaly-based intrusion detection techniques, are in general neural network models with a large number of hidden layers. These models can learn extremely complicated non-linear functions, and their hierarchical layer structure allows them to acquire meaningful feature representations from incoming data. Researchers have explored deep learning techniques for in-vehicle intrusion detection on Controller Area Network (CAN) bus protocol since 2015 [125]. However, due to

the lack of relevant and public datasets, few studies have been conducted to study the intrusion detection performance of deep learning based IDS for automotive systems using Automotive Ethernet-based network. Among them, Alkhatib et al. [14] proposed a deep learning-based sequential model for offline intrusion detection on Scalable Service-Oriented Middleware over IP (SOME/IP) application layer protocol on top of Automotive Ethernet. Moreover, Jeong et al [74] presented an intrusion detection method for detecting audio-video transport protocol (AVTP) stream injection attacks in Automotive Ethernet-based networks.

In this chapter, we compare the performance of different deep and machine learning based intrusion detection systems for real-time detection of anomalies on the AVTP protocol. Regarding deep learning based models, we leverage different types of autoencoders which reconstructs a sequence of exchanged AVTP packets over the in-vehicle network. Anomalies in AVTP packet stream, which may lead to critical interruption of media streams, are therefore detected by computing the corresponding reconstruction error. These models are compared with other state-of-the-art anomaly detection models such as One-class SVM (OCSVM), Local Outlier Factor (LOF), and Isolation Forest. The numerical results, conducted on the recently published **"Automotive Ethernet Intrusion Dataset"**, show that deep learning based models outperform other baselines under different experimental settings.

The main contributions of this paper are as follows:

- We compare the performance of different unsupervised anomaly detection methods to detect unknown cyberattacks in real-time on AVTP protocol used in Automotive Ethernet-based in-vehicle network for media streaming.
- We evaluate their performance by using the recently published "Automotive Ethernet Intrusion Detection" dataset [72] and which contains replay attacks.

This chapter is organized as follows. In Section 7.2, we present an overview of media stream transportation using AVTP network protocol. In Section 7.3, we present an overview of the considered AVTP dataset, the covered threat model along with the engendered cyberattacks. Section 7.4 discusses the detection of in-vehicle network anomalies using unsupervised anomaly detection algorithms. We discuss our experimental results in Section 7.5.

7.2 Transmission of Media Streams using AVTP

Traditional in-vehicle networks are mostly based on bus technology that can not keep up with the growing communication demands of self-driving cars. In fact, they cannot meet the in-vehicle

network requirements for high bandwidth, reliability, and real-time communication expectations. Automotive Ethernet, a novel in-vehicle network communication technology, is implemented to ensure an appropriate level of quality of service (QoS) which is essential for time-critical automotive applications.

Audio Video Bridging (AVB) over Ethernet, a set of technical standards, provides improved synchronization, low latency, and reliability for switched Ethernet networks between multimedia devices. Recently, a lot of automotive products such as end-nodes devices (i.e., speakers, cameras, digital signal processors) and a network hub (i.e., AV Bridges) support it. In fact, end nodes can be a talker, a listener or both. A talker is the transmitter of a data stream or the source of the AVB stream and a listener is the receiver or the destination of the AVB stream. These end nodes are connected by an AVB Bridge which acts as a switch that receives time-critical data from the AVB talker and forwards it to the AVB listener. This interconnection between these three components, as presented in Fig.7.1, is called AVB Ethernet Local Area Network (LAN).

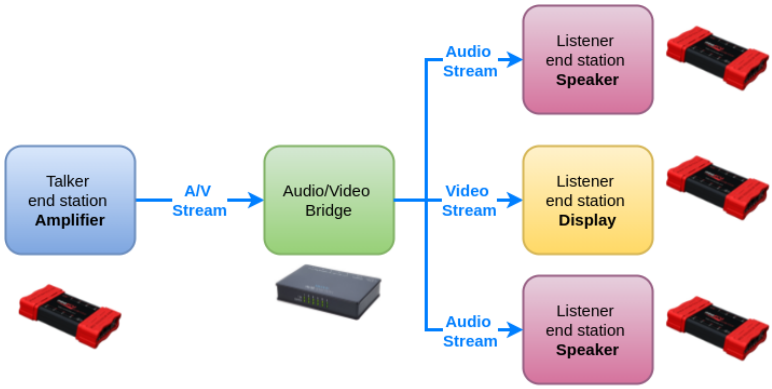

Figure 7.1: Typical AVB Ethernet Local Area Network (LAN).

As previously mentioned, AVB has diverse sub-standards to support time-critical in-vehicle applications such as IEE 802.1 Qav, IEEE 802.1 Qat, IEEE 802.1 AS, and IEEE 1722. Due to the lack of publicly available datasets which cover attacks on diverse AVB protocols, we are only considering published ones that are composed of captured automotive cyberattacks on IEEE 1722, a stream transmission protocol in charge of transporting control data and audio and video streams. Unfortunately, datasets which cover attacks on other AVB protocols aren't publicly available. As depicted in Fig. 7.2, the IEEE 1722 packet and its content are sent through an Ethernet frame. The IEEE 802.1Q header is also included in the Ethernet packet. Furthermore, the priority information encapsulated within is critical for the functioning of the AVB QoS

concept. Moreover, only AVB listener members that share the same AVB talker's VLAN tag can receive the audio/video stream. In the case of IEEE 1722, the ethertype field's hexadecimal value is 0X22F0.

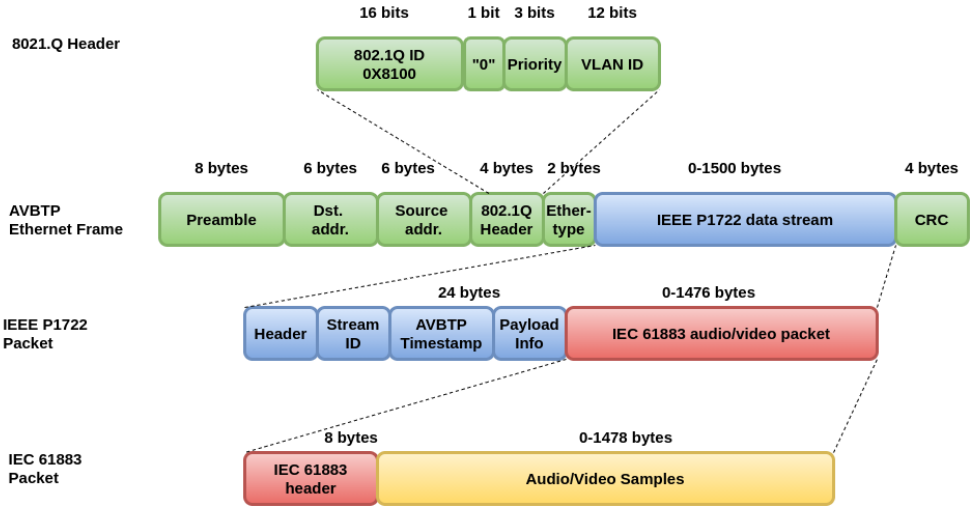

Figure 7.2: IEEE 1722 packet format. Source: [100]

In terms of IEEE 1722 streaming packets, the header, the stream ID, the "Presentation time," payload information, and the payload itself are all included therein. The data type of the A/V stream is specified in the header which also includes its sequence number needed by AVB listeners to detect missing packets. The MAC address of the talker is used to produce the stream ID, which identifies a single data stream. The format of the data within the payload is directly related to the field of the payload information. The AVBTP timestamp is a time presentation which specifies when a received packet should be delivered to the AVB listener application [100].

We will provide in Section 7.3 the threat model and the corresponding replay attacks on AVTP protocol, created by Jeong et al. [74], and list also the relevant AVTP features to be leveraged for anomaly detection.

7.3 AVTP Dataset Description

Given a sequence of AVTP packets, we aim to detect whether this sequence is normal or anomalous, i.e., an AVTP sequence is anomalous if it contains at least one abnormal (i.e., injected/out of order/replayed) packet. Hence, we have used the **"Automotive Ethernet**

Intrusion Dataset" dataset [72] created by Jeong et al. [74] and which contains benign and malicious AVTP packet captures from their physical Automotive Ethernet testbed.

Dataset	# Normal packets	# Abnormal packets	Size (MB)
$\mathcal{R}$	0	36	0.0164
$\mathcal{D}_{normal}$	139,440	N/A	63.3
$\mathcal{D}^1_{injected}$	139,440	65,988	93.3
$\mathcal{D}^2_{injected}$	307,020	130,906	198.8

Table 7.1: Automotive Ethernet Intrusion Dataset

The datasets are recorded in the PCAP file format and, therefore, are viewed using prevalent programming libraries and packet analyzers (such as Wireshark). In fact, the dataset contains four benign (attack-free) packet captures and four malicious ones collected in different environments. The malicious packet captures represent **replay cyberattack**. In fact, they contain **message injection** of arbitrary stream AVTP data units (AVTPDUs) into the IVN since the attacker's goal is to output a single video frame, at a terminal application connected to the AVB listener, by injecting previously generated AVTPDUs during a certain period. For our experiment, we have only considered the AVTP packets collected indoor, presented in Table 7.1. We refer readers to [74] for further information.

In order to represent AVTP sequences, we use the **Feature-based Sliding Window (FSW)**[145] to group packets which belong to an AVTP dataset into subsequences with fixed window size $\mathbf{w}$, where $\mathbf{w} \in \{8, 16, 24, 32, 40\}$ and the slide size is 1. Hence, each sequence of ordered packets is defined as $S = \{\mathbf{p}_1, ..., \mathbf{p}_t, ..., \mathbf{p}_T\}$, where $\mathbf{p}_t \in \mathcal{D}$ indicates a transmitted AVTP packet at time t, and $\mathcal{D}$ indicates the original AVTP Dataset. Each packet $\mathbf{p}_t$ in the AVTP dataset has 438 bytes/features, each of which has a integer value between 0 and 255, where $\mathbf{p}_t \in \mathbb{Z}^{58}$ (since the most suitable number of bytes used to detect anomalies is the first 58 bytes of each AVTP packet,[74]). Hence, to achieve our previously mentioned goal, we train our model using the dataset $\mathcal{D}_{training}$ composed of normal AVTP sequences with each packet. The normal sequences are extracted from dataset $\mathcal{D}_{normal}$, depicted in Table 7.1. However, when testing, we have preprocessed packets into sequences from both datasets $\mathcal{D}^1_{injected}$ and $\mathcal{D}^2_{injected}$, and which contain replayed packets from dataset $\mathcal{R}$. Moreover, we label each AVTP sequence using the following criteria:

$$Y = \begin{cases} 0 \text{ (normal)} & \text{if } (\mathbf{p}_t \in S) \& (\mathbf{p}_t \notin \mathcal{R}), \forall t \in \{1, .., w\} \\ 1 \text{ (abnormal)} & \text{otherwise} \end{cases}$$

where Y is an AVTP sequence's label, and $\mathcal{R}$ is a set of replayed AVTP packets, collected during a legitimate AVTP media transmission.

It's worth noting that we do not follow [74] labeling criteria. In fact, [74] aim to detect packets which are replayed. However, we aim to detect whether a sequence contains one or several injected packets. Notably, using this labeling criteria, more suitable for self-supervised learning, our model can be further used for the detection of cyberattacks different than replay attacks in future work and which are detected by inspecting a series of ordered packets.

Moreover, we have reshaped our dataset to suit different types of models. Hence, since convolutional autoencoders, presented in Section 7.4, deal with image samples we had to reshape each sequence S into 2D images using the following mapping

$$Im(S_k) = \begin{pmatrix} a_{k,1} & a_{k,2} & \cdots & a_{k,58} \\ a_{k+1,1} & a_{k+1,2} & \cdots & a_{k+1,58} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k+w,1} & a_{k+w,2} & \cdots & a_{k+w,58} \end{pmatrix} \tag{7.1}$$

where $Im(S_k)$ denotes the k-th reshaped sequence of S (corresponding to training sample $k \in \{1, ..., N\}$), w is the total sequence length, $a_{m,n}$ is an AVTP packet feature (byte) such that $0 \leq a_{m,n} \leq 255$, $k \leq m \leq k + w$, and $1 \leq n \leq 58$. Hence, an AVTP dataset, represented as $\mathcal{D} = \{Im(S_k)\}_{k=1}^{N}$, is ready to be fed into the CAE model. It's worth noting, that when fed into LSTM models, we use $\mathcal{D} = \{S_k\}_{k=1}^{N}$, where S_k is defined as the kth AVTP sequence.

7.4 Unsupervised Intrusion Detection Systems

Intrusion Detection Systems (IDSs) are considered as an efficient tool to guarantee the confidentiality, integrity and availability of network data. In fact, network intrusions can be detected and identified by comparing their attack signatures to a dataset which contains a pre-defined list of cyberattack patterns. This approach is called signature-based intrusion detection. However, a regular updating of signature databases is not practicable because of the constant evolution of innovative attack tactics. An alternative solution could be the adoption of anomaly-based IDSs which find pattern in the data that deviates from other observations and indicates the presence of malicious activities in the network traffic. In this work, we will compare the performance of deep learning based intrusion detecton systems especially autoencoders based models with state-of-the-art machine learning models.

7.4.1 Deep Learning-based IDS

Deep learning techniques are increasingly used to address the development of complex anomaly detection based IDSs [37]. One of the most commonly studied feature learning techniques is the use of autoencoder (AE) neural networks which can be used to detect anomalies in high-dimensional data and for different data types including images/videos, sequence data and graph data.

The autoencoder AE, introduced by Rumelhart et al. [115], seeks to learn a low-dimensional feature representation space suitable for reconstructing the provided data instances, as explained in Chapter 2. Essentially, only data with normal instances are used to train the AE. Hence, since normal samples in the test dataset have likewise normal profile of training samples, the corresponding reconstruction error is alike. However, compared to the anomalous testing samples, the reconstruction error is much higher. As a result, we can simply classify samples by defining a threshold for reconstruction error:

$$c(\mathbf{x}) = \begin{cases} 0 \text{ (normal)} & s_x < \beta \\ 1 \text{ (abnormal)} & s_x > \beta \end{cases} \tag{7.2}$$

where c($\mathbf{x}$) is the classification function for input sample $\mathbf{x}$ and β is the pre-defined anomaly detection threshold.

Parameter	**Value**
Learning Rate	0.0001
Optimizer	Adam
Batch Size	16
Early stopping	Yes

Table 7.2: AE Models Configuration

Through our work, we will investigate the performance of two types of autoencoders: Convolutional based autoencoder (CAE), and Long Short Term Memory based autoencoder (LSTMAE), covered in 2. To implement these models, we leverage the Python deep learning framework Pytorch [13]. We train and evaluate them on NVIDIA® Tesla® V100S with 32 GB HBM2 memory. After hyperparameter tuning, we use the commonly chosen hyperparameters depicted in Table 7.2.

7.4.1.1 Convolutional based Autoencoder (CAE)

For different sequence lengths w, we have developed different CAE architectures. Our CAE architecture, depicted in Table. 7.3, is composed of three convolutional layers on the encoder side, flatten and unflatten layers, one embedding layer, and three deconvolutional layers on the decoder side. For the encoding module, we first stack three convolution layers with 36, 64, and 128 feature maps, respectively. We have chosen 3x3 kernel sizes for the different convolutional layers and set the padding and the stride is set to (1,1) and (2,2), respectively. Then we flatten the output of the encoder and feed it to a dense layer that represents the latent space and which is composed of 64 * w neurons (chosen after tuning the correspondent number of neurons). The embedded vector is then unflattened and fed into the decoder. As for the decoding module, we flip the architecture of the encoder, i.e., the corresponding feature maps from bottom to up are 128, 64,32, and 1, and the kernel sizes are 3×3. We set the stride to (1,1), the padding to (2,2), and the output padding to (1,0), (1,0), and (1,1) for the three deconvolutional layers.

Block	Layer	Dimensions	Act. Function	Filter Size	Stride	Padding	Output Padding
-	Input	(1,w,58)	-	-	-	-	-
Encoder	Conv1	(32,w/2,29)	ReLU	(3,3)	(1,1)	(2,2)	-
	Conv2	(64,w/4,15)	ReLU	(3,3)	(1,1)	(2,2)	-
	Conv3	(128,w/8,8)	ReLU	(3,3)	(1,1)	(2,2)	-
	Flatten	(128*w)	ReLU	(3,3)	-	-	-
Embedding	Linear	(128*w/2)	ReLU	-	-	-	-
Decoder	Unflatten	(128,w/8,8)	ReLU	-	-	-	-
	Deconv1	(64,w/4,15)	ReLU	(3,3)	(1,1)	(2,2)	(1,0)
	Deconv2	(32,w/2,29)	ReLU	(3,3)	(1,1)	(2,2)	(1,0)
	Deconv3	(1,w,58)	ReLU	(3,3)	(1,1)	(2,2)	(1,1)

Table 7.3: CAE's Model Architecture

7.4.1.2 Long Short Term Memory based Autoencoder

For each sequence length w, we create a different LSTMAE. As shown in Table 7.4, for the LSTM based encoding module, we firstly stack two LSTM layers which output an embedding vector of size 10 (chosen after tuning). Then we repeat the embedding vector w times, and feed it into the decoder. As for the decoding module, we flip the architecture of encoder, i.e. the repeated vector passes through two LSTM layers with a number of features 10 and 20 respectively and a dense layer, to be finally reconstructed.

Block	Layer	Output Dimensions	Activation Function
-	Input	(w,58)	-
Encoder	LSTM1	(w,20)	ReLU
Embedding	LSTM2	(1,10)	ReLU
Decoder	Repeat	(w,10)	ReLU
	LSTM1	(w,10)	ReLU
	LSTM2	(w,20)	ReLU
	Linear	(w,58)	-

Table 7.4: LSTMAE's Model Architecture

7.4.1.3 Anomaly Detection using AE models

As previously mentioned, we will classify AVTP sequence samples by defining a threshold β. Hence, after training our AE models for each window size w, we vary β between $\mu - \alpha_{min}\sigma$ and $\mu + \alpha_{max}\sigma$ where μ is the mean reconstruction error of normal samples used for training, σ is the standard deviation of normal samples' reconstruction errors, $\alpha \in \{-2, 2\}$ with a step size $\delta = 0.5$, $\alpha_{max} = max(\alpha)$ and $\alpha_{min} = min(\alpha)$ to select the best threshold.

7.4.2 Machine Learning-based IDS

Through our work, we compare the autoencoder-based models to state-of-the-art machine learning-based anomaly detection algorithms: One-Class SVM (OCSVM), Local Outlier Factor (LOF) and Isolation Forest (IF), explained earlier in Chapter 2. We have implemented these algorithms using the Scikit-learn python library, and have trained and evaluated them on a 3.3 GHz AMD EPYC™ 7402.

7.5 Results

To assess the performance of different anomaly-based IDS, we use the F1-score metric explained in Chapter 2. Figures 7.3 and 7.4 shows the performance of different anomaly based IDS on both datasets $\mathcal{D}^1_{injected}$ and $\mathcal{D}^2_{injected}$. As seen, conventional machine learning algorithms such as OCSVM, Isolation Forest, and Local Outlier Factor perform poorly on both datasets when recognizing anomalous AVTP sequences for different sequence length. In fact, these traditional anomaly detection models are inefficient at detecting anomalies in large, high-dimensional datasets since these methods assume small datasets with low numbers of features. Hence, when dealing with a huge input dimensionality, a high proportion of irrelevant features can effectively

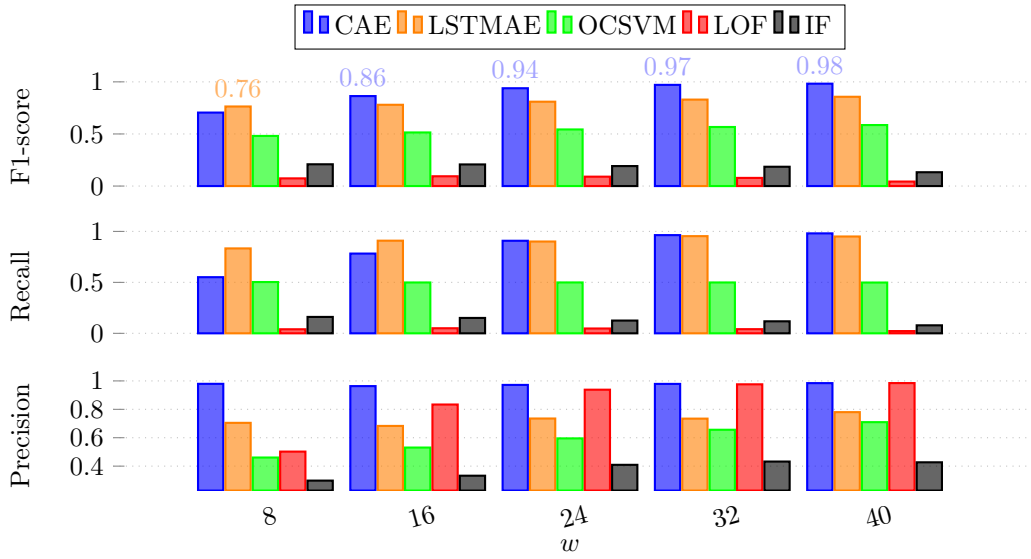

Figure 7.3: Comparision of different unsupervised machine learning anomaly detection performance under different window sizes w on $\mathcal{D}^1_{injected}$

creates noise in the input data, which masks the true anomalies and engenders poor anomaly detection performance.

To overcome the limitations of these approaches in high-dimensional datasets, the deep learning models CAE and LSTMAE, are considered as a better alternative for anomaly detection. As demonstrated in Figures 7.3 and 7.4, they significantly outperform the benchmark anomaly detection models and achieve reasonbale F1-scores on both datasets. After tuning the threshold β for various sequence length and for different datasets and AE models, the CAE and LSTMAE reached their highest performance when $\beta = \mu + 0.5\sigma$. Moreover, the CAE model achieves an overall better performance in terms of F1-score scores than LSTMAE model which indicates that LSTMAE is not able to encode the context information of an AVTP sequence from both the left and right context especially when working on long sequences ($w \geq 16$). Despite the fact that a Bidirectional LSTMAE is commonly used nowadays to represent contextual information, they suffer from the vanishing or exploding gradients. In other words, the model hardly captures the long-term dependency and which is critical for the detection of anomalies in large sequences. When varying the AVTP sequence length between 16 and 40, CAE has outperformed LSTMAE by exploiting significant correlations in a sequence of AVTP packets. The performance of both models proportionally increases when increasing window length on both datasets, since AVTP sequences will contain more injected packets, thus it becomes easier to differentiate between

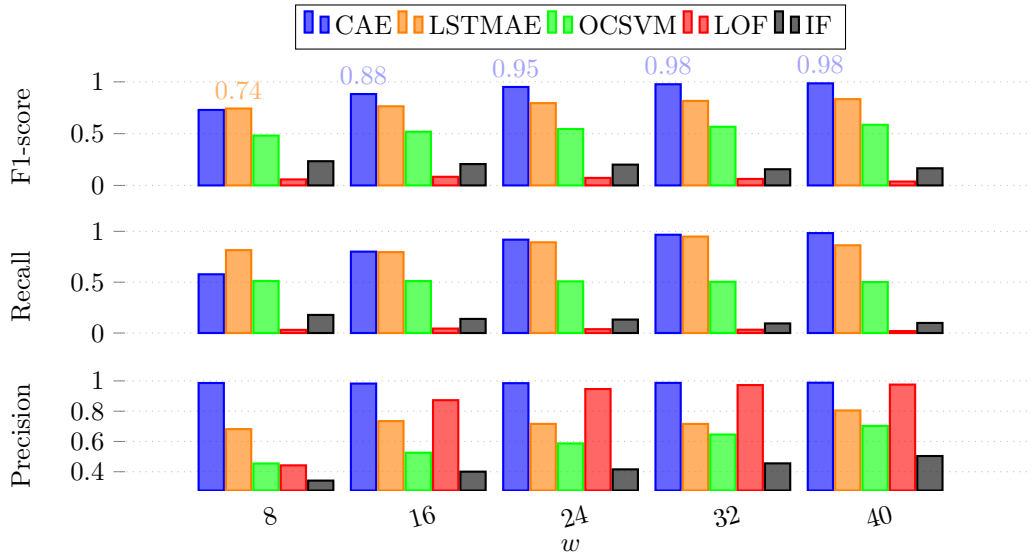

Figure 7.4: Comparison of different unsupervised machine learning anomaly detection performance under different window sizes w on $\mathcal{D}^2_{injected}$

normal and abnormal AVTP sequences. We also assess the performance of the best models, more

Window	Conv-AE			LSTM-AE		
	Inference Time (ms)	#Parameters	Model Size (MB)	Inference Time (ms)	#Parameters	Model Size (KB)
8	0.49 ± 0.53	1,235,329	4.8	1.17 ± 0.90	12,338	52
16	0.48 ± 0.52	4,382,593	17	1.58 ± 0.96	12,338	52
24	0.38 ± 0.33	9,627,009	37	1.91 ± 0.95	12,338	52
32	0.43 ± 0.28	16,968,577	65	2.31 ± 1.02	12,338	52
40	0.45 ± 0.14	26,407,297	101	2.70 ± 0.99	12,338	52

Table 7.5: AutoEncoder-based Models' Characteristics & Computational Resources

specifically AE models, by measuring their computational power and their memory requirements. As depicted in Table 7.5, although the CAE model has a bigger number of parameters and a larger model size than LSTMAE, it stays speedier when detecting anomalies in AVTP sequences for different window sizes. Hence, CAE is more suitable for real-time intrusion detection than LSTMAE. Although it has a larger model size, the CAE models can either be deployed on a cloud server connected to the in-vehicle network or can be embedded inside an ECU with suitable memory characteristics. In the future, we plan to examine the implementation of both ideas.

7.6 Conclusion

We compared the performance of different deep and machine learning algorithms for learning normal AVTP communication behavior and thus identifying cyberattacks on this protocol. Our experimental results confirmed the prominence of autoencoder-based IDS over traditional machine learning models for different AVTP sequence lengths. For real-time detection, we suggested convolutional-based AE, whose model complexity is more suitable than LSTM-AE. However, as the evaluation dataset only covered replay attacks, our current comparison of semi-supervised deep learning and unsupervised machine learning needs further assessment once the different types of cyberattacks on the AVTP protocol are available. Additionally, the compared techniques modeled normal communications between two AVB Talkers, i.e., the legitimate talker and the attacker, and a single AVB listener. Therefore, it's crucial to evaluate if the proposed deep learning techniques - already trained - can generalize to new test datasets in which AVTP traffic for a larger in-vehicle network with more AVB listeners and talkers is exchanged. Finally, this problem also motivates us to design more powerful neural network structures. Recently, researchers are addressing similar issues using dynamic neural networks [56]. Compared to static models which have fixed computational graphs and parameters at the inference stage, dynamic networks can adapt their structures or parameters to different inputs, leading to notable advantages in terms of accuracy, computational efficiency, adaptiveness, etc. Considering the dynamic nature of the AVB network, different representative inputs may have diverse computational demands for the devised IDS. Hence, it is natural to perform inference with dynamic architectures conditioned on each sample. Specifically, one can adjust the network depth and width. Networks with dynamic architectures not only save redundant computation for canonical ("easy") samples, but also, preserve their representation power when recognizing non-canonical ("hard") samples.

Part IV

Conclusions and Perspectives

CHAPTER 8

Conclusions and Perspectives

8.1 Conclusion

The problem of in-vehicle intrusion detection needs to be urgently addressed, where it is desirable to determine known and novel automotive attacks. As the E/E architecture of the vehicle is always evolving to meet the consumers' demands, hackers are always finding ways to attack existing and recently deployed automotive protocols. Throughout this thesis, we investigate Deep Learning as a solution to build different intrusion detection techniques that are mapped to various protocols including CAN and Automotive Ethernet, and for different intrusion tasks such as classification and anomaly detection. For the CAN-based IVN, we particularly investigated the ability of the attention-based model "BERT" to detect targeted in-vehicle intrusions in chapters 3 and 4. Inspired by BERT's success in numerous natural language processing tasks (NLP) and its powerful expressive capability, we leverage its attention-based architecture to model non-NLP tasks, particularly when modeling the normal exchange of CAN packets between ECUs and the detection of sequential intrusions on the CAN bus. In Chapter 3, we investigate whether BERT can be regarded as a frequency-based method and thus able to detect attacks that result in fast message timing or the new appearance of new CAN IDs, and slow message timing or the disappearance of usually present IDs. Drawing on the prominent findings, we further investigate in Chapter 4 its potential in detecting also stealthier attacks that do not disrupt normal timing or ID distributions, but rather the interrelations between asynchronously

transmitted CAN signals. However, since securing CAN isn't enough to ensure in-vehicle network security as different automotive protocols have been developed, we have further studied other critical networking systems. Particularly, we focused on the Automotive Ethernet protocol as it is a flexible and scalable networking technology in communication systems and is recently standardized and adopted for in-vehicle communication. For the classification task, as presented in Chapter 5, we have successfully contributed to developing an offline RNN-based model that monitors the transmitted SOME/IP network packets between different clients and servers and classifies the network traffic into corresponding attack types or the normal class. As there is a lack of available datasets for the SOME/IP protocol, we generated and labeled our own which was thus used to train the proposed model using supervised learning techniques and for its evaluation. The main purpose behind our contribution is to assess the RNN's ability in discovering specific intrusions on the SOME/IP protocol (anomalies of interest). Concerning the detection of attacks, we leveraged both signature-based and anomaly-based approaches. In Chapter 6, we tackle the high dimensionality challenge of the in-vehicle network traffic. We have thus considered the problem of intrusion detection for SOME/IP protocol and presented "SAID" a novel technique for the detection of anomalies from a large sequence of exchanged SOME/IP network packets. The proposed detector leverages a self-attention-based neural network to model the contextual dependencies between SOME/IP packets. A comparative study with various deep learning algorithms is performed to show the outstanding performance of our proposed detector in quality while being more parallelizable and requiring significantly less time to detect intrusions. In Chapter 7, we leveraged self-supervised and unsupervised learning techniques for anomaly detection for the ethernet-based protocol AVTP as the previously conducted research studies were only dedicated to the detection of replay attacks using supervised learning techniques. By studying the performance of several autoencoders and comparing their efficiency to classical machine learning techniques, we contributed to building a comprehensive and real-time IDS that is not limited to the detection of specific attacks but rather leveraged for the detection of zero-day AVTP attacks also.

8.2 Contribution improvements

In Chapters 5 and 6, we presented two supervised learning-based techniques for the detection of intrusions on SOME/IP protocol. An improvement of this contribution consists of developing a complementary unsupervised learning-based technique for the detection of novel attacks

once a richer dataset with a richer representation of the normal SOME/IP communication behavior is publicly available. Additionally, since all our contributions regarding in-vehicle intrusion detection are conducted on different datasets, we seek to group them in one general framework that monitors the automotive protocols altogether. Finally, we have considered the development of in-vehicle IDS without taking into consideration the availability of other security countermeasures including authentication protocols, firewalls, honeypots, etc. To eventually improve our contribution, we should also detect stealthier automotive intrusions against a more secure in-vehicle environment.

8.3 Further perspectives

8.3.1 Distributed IDS (dIDS)

Within the scope of this thesis, we have suggested the implementation of a centralized IDS for several in-vehicle network protocols. However, we believe that there are significant advantages to devising a distributed IDS system over a centralized one. In fact, as the E/E architecture of the vehicle is regarded as a distributed system, no central ECU is knowledgeable of all security events on the other ECUs making the establishment of a distributed IDS necessary. Several other factors are decisive for this proposal including the avoidance of a single point of failure from a security perspective and the automotive-specific constraints of the ECUs, such as limited computing power and low memory capacity. In this case, in-vehicle network IDS sensors monitor data traffic for specific network segments and stay alert for any suspicious activity in network, transport, and application protocols [8].

8.3.2 Neural network Quantization

Recent research on Neural Networks (NNs) models for in-vehicle intrusion detection has mostly focused on enhancing the intrusion detection rate while simultaneously reducing the number of false alarms (false positive rate). Although the performance of these over-parameterized (and therefore very large) neural network models has considerably improved, the sheer size of these models prohibits their deployment on resource-constrained electronic control units (ECUs) [53]. For future work, it's necessary to devise a deep learning-based intrusion detection system (IDS) that predicts attacks on the in-vehicle network in real-time while providing an optimal trade-off between accuracy and resource consumption. Thus, an investigation regarding **"Quantization"**, an approach that has shown great and consistent success in both *training* [30, 38] and *inference*

of NN models, must be considered. In quantization, the goal is to reduce the precision of both the model's learnable parameters - stored in floating point precision - as well as the intermediate activation values to low precision, with minimal impact on the generalization power/accuracy of the model. Several quantization methods have thus been proposed to map the floating point value to a quantized one including simulated and integer-only [70], mixed-precision [45], hardware aware [60], distillation-assisted [76], extreme [75], and vector quantization [50].

By conducting a comparative analysis between the aforementioned quantization methods, future work must be concerned about checking which method respects mostly the tight resource constraints of ECUs including compute, memory, and power budget without significantly impacting the intrusion detection accuracy of the IDS. Additionally, although there are several hardware platforms in the context of quantization, no previous research work has been conducted to study the quantization approach for NN-based IDS on specific types of automotive microcontrollers or ECUs [3] including the 32-bit SPC5 family built on Power Architecture, the 8-bit STM8A family as well as 16-bit ST10 legacy MCUs and others. For general use cases, the STM32 (32-bit RISC ARM processor cores), a family of microcontrollers based on the ARM Cortex-M cores, was used for NN inference at the edge. Because some of the ARM Cortex-M cores do not include dedicated floating-point units, the models should first be quantized before deployment. CMSIS-NN [83] is a library from ARM that helps quantize and deploy NN models onto the ARM Cortex-M cores. Specifically, the library leverages fixed-point quantization with power-of-two scaling factors so that quantization and dequantization processes can be carried out efficiently with bit-shifting operations. Inspired by this success, a study must be focused on the comparison of the throughput of different commercial automotive ECUs for NN inference based on the adopted quantization approach.

8.3.3 Explainable IDS (X-IDS) for IVNs

Automotive security analysts heavily rely on the IDS systems to make decisions about the detected IVN threats for a variety of purposes including alert escalation, threat and attack mitigation, intelligence gathering, and forensic analysis among others. However, IDSs designed using deep learning techniques are often treated as black box models and do not provide a justification for their prediction. This creates a barrier for security analysts, as they are unable to improve their decisions based on the model's predictions. To address this issue, researchers have started to design explainable IDS (X-IDS) by leveraging explainable artificial intelligence (XAI) techniques for IDS [119]. Researchers have started recently to build such trustworthy IDS

using XAI techniques for IVNs [98]. We believe that these techniques can build operator trust and allow for more control of autonomous AI systems.

8.3.4 Adversarial Robust IDS for IVNs

Despite the tremendous advantage of deep learning techniques in learning underlying threat patterns and features, they are susceptible to attacks, i.e., attacks wherein slight perturbations of the input features cause misclassifications. Hence, as soon as the proposed IDS models are deployed in today's modern vehicles or their dedicated vehicular security operation center (V-SOC), they become targets of attacks that severely undermine their capability, and even turn them into unconventional attack tools. In fact, several attacks can exploit the IDS vulnerabilities and thereby be conducted against deep learning-based IDS including evasion, overstimulation, poisoning, Denial Of Service (DoS), response hijacking and reverse engineering [40]. Although numerous research studies are investigating the adversarial attacks against deep learning-based techniques [94, 132, 133, 137] and the corresponding defensive methods in IDS [94, 111, 126] for computer networks, few research work is achieved particularly for the IVN domain [92].

Bibliography

[1] How software is eating the car. https://www.youtube.com/watch?v=g5MezxMcRmk&t=7s. 4

[2] Automotive Intrusion Detection Systems. https://www.vector.com/fr/fr/connaissances/security/automotive-intrusion-detection-systems/. 28

[3] Automotive Microcontrollers. https://www.st.com/en/automotive-microcontrollers.html. 130

[4] AVTP Protocol Specification. https://avnu.org/wp-content/uploads/2014/05/AVnu-AABAC_IEEE-1722-Media-on-AVB-Networks_Rob-Silfvast.pdf. 27, 112

[5] CIC DoS dataset (2017). https://www.unb.ca/cic/datasets/dos-dataset.html. 13

[6] CSE-CIC-IDS2018 on AWS. https://www.unb.ca/cic/datasets/ids-2018.html. 13

[7] DoIP Protocol Specification. https://www.iso.org/obp/ui/#iso:std:iso:13400:-2:ed-2:v1:en. 27

[8] Intrusion detection as a distributed system. https://www.etas.com/en/company/news-intrusion-detection-as-a-distributed-system.php. 129

[9] Intrusion Detection Evaluation Dataset (CIC-IDS2017). https://www.unb.ca/cic/datasets/ids-2017.html. 13

[10] ISO 11898-2:2016 - Controller area network (CAN) — Part 2: High-speed medium access unit. https://www.iso.org/standard/67244.html. 20

[11] ISO/SAE 21434:2021 Road vehicles — Cybersecurity engineering. https://www.iso.org/standard/70918.html. 28

[12] Keras. https://keras.io. 89

[13] Pytorch framework:. https://pytorch.org/. 10, 57, 70, 118

[14] RNN-based IDS for SOME/IP Intrusion Detection. https://github.com/Alkhatibnatasha/SOMEIP_IDS. 82, 97, 98, 113

[15] Scapy [Online] Available:. https://www.etas.com. 112

[16] SOME/IP Generator [Online] Available. https://github.com/Egomania/SOME-IPGenerator. 82, 101

[17] SOME/IP Protocol Specification. https://www.autosar.org/fileadmin/standards/foundation/1-4/AUTOSAR_PRS_SOMEIPProtocol.pdf. 26, 78, 81

[18] SynCAN Dataset. https://github.com/etas/SynCAN. 70

[19] Tensorflow framework:. https://www.tensorflow.org/. 10

[20] The history of Ethernet. https://www.youtube.com/watch?v=g5MezxMcRmk&t=7s. 21

[21] UDP-NM Protocol Specification. https://www.autosar.org/fileadmin/standards/classic/19-11/AUTOSAR_SWS_UDPNetworkManagement.pdf. 27

[22] UN Regulation No. 155 - Cyber security and cyber security management system. https://unece.org/transport/documents/2021/03/standards/un-regulation-no-155-cyber-security-and-cyber-security. 8

[23] UN Regulation No. 156 - Software update and software update management system. https://unece.org/transport/documents/2021/03/standards/un-regulation-no-156-software-update-and-software-update. 8

[24] Ieee standard for ethernet. *IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012)*, pages 1–4017, 2016. 21

[25] Lina Achaji, Thierno Barry, Thibault Fouqueray, Julien Moreau, Francois Aioun, and Francois Charpillet. Pretr: Spatio-temporal non-autoregressive trajectory prediction transformer. *arXiv preprint arXiv:2203.09293*, 2022. 65, 67

[26] Arwa Aldweesh, Abdelouahid Derhab, and Ahmed Z Emam. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, 189:105124, 2020. 11

[27] Natasha Alkhatib, Maria Mushtaq, Hadi Ghauch, and Jean-Luc Danger. Can-bert do it? controller area network intrusion detection system based on bert language model. *arXiv preprint arXiv:2210.09439*, 2022. 65, 99

[28] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6836–6846, 2021. 65, 67

[29] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 52

[30] Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. Scalable methods for 8-bit training of neural networks. *Advances in neural information processing systems*, 31, 2018. 129

[31] Lucia Lo Bello. The case for ethernet in automotive communications. *ACM SIGBED Review*, 8(4):7–15, 2011. 112

[32] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19, 2006. 10

[33] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, volume 2, page 4, 2021. 65

[34] Andrea Busnelli. Car software: 100m lines of code and counting." linkedin. com. Technical report, accessed 2015-8-5, 2014. 81

[35] Zhiqiang Cai, Aohui Wang, Wenkai Zhang, M Gruffke, and H Schweppe. 0-days & mitigations: roadways to exploit and secure connected bmw cars. *Black Hat USA*, 2019:39, 2019. 6

[36] Andreas Casparsen, Daniel Greth Sϕrensen, Jeppe Nellemann Andersen, Jonas Ingerslev Christensen, Panagiotis Antoniou, Rolf Krϕyer, Tatiana Madsen, and Karsten Gjoerup. Closing the security gaps in some/ip through implementation of a host-based intrusion detection system. In *2022 25th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 436–441. IEEE, 2022. 98

[37] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019. 118

[38] Brian Chmiel, Liad Ben-Uri, Moran Shkolnik, Elad Hoffer, Ron Banner, and Daniel Soudry. Neural gradients are near-lognormal: improved quantized and sparse training. *arXiv preprint arXiv:2006.08173*, 2020. 129

[39] Jun Kang Chow, Zhaoyu Su, Jimmy Wu, Pin Siang Tan, Xin Mao, and Yu-Hsing Wang. Anomaly detection of defects on concrete structures with the convolutional autoencoder. *Advanced Engineering Informatics*, 45:101105, 2020. 39

[40] Igino Corona, Giorgio Giacinto, and Fabio Roli. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences*, 239:201–225, 2013. 131

[41] Yu Cui, Yiping Sun, Jinglu Hu, and Gehao Sheng. A convolutional auto-encoder method for anomaly detection on system logs. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3057–3062. IEEE, 2018. 39

[42] Dorothy E Denning. An intrusion-detection model. *IEEE Transactions on software engineering*, (2):222–232, 1987. 12

[43] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 50, 65, 67, 69

[44] Nguyet Quang Do, Ali Selamat, Ondrej Krejcar, Enrique Herrera-Viedma, and Hamido Fujita. Deep learning for phishing detection: Taxonomy, current challenges and future directions. *IEEE Access*, 2022. 11

[45] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302, 2019. 130

[46] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 99

[47] Jinze Du, Rui Tang, and Tao Feng. Security analysis and improvement of vehicle ethernet some/ip protocol. *Sensors*, 22(18):6792, 2022. 96, 98

[48] Tobias Gehrmann and Paul Duplys. Intrusion detection for some/ip: Challenges and opportunities. In *2020 23rd Euromicro Conference on Digital System Design (DSD)*, pages 583–587. IEEE, 2020. 80, 81, 97, 98

[49] Mabrouka Gmiden, Mohamed Hedi Gmiden, and Hafedh 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. IEEE, 2016. 79

[50] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014. 130

[51] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. xix, 9, 10

[52] Haixuan Guo, Shuhan Yuan, and Xintao Wu. Logbert: Log anomaly detection via bert. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021. 50

[53] Yunhui Guo. A survey on methods and theories of quantized neural networks. *arXiv preprint arXiv:1808.04752*, 2018. 129

[54] Mee Lan Han, Byung Il Kwak, and Huy Kang Kim. Anomaly intrusion detection method for vehicular networks based on survival analysis. *Vehicular communications*, 14:52–63, 2018. 80

[55] Mee Lan Han, Byung Il Kwak, and Huy Kang Kim. Tow-ids: Intrusion detection system based on three overlapped wavelets for automotive ethernet. *IEEE Transactions on Information Forensics and Security*, 18:411–422, 2022. 61

[56] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 123

[57] Peter Hank, Steffen Müller, Ovidiu Vermesan, and Jeroen Van Den Keybus. Automotive ethernet: in-vehicle networking and smart mobility. In *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1735–1739. IEEE, 2013. 22

[58] Peter Hank, Steffen Müller, Ovidiu Vermesan, and Jeroen Van Den Keybus. Automotive ethernet: in-vehicle networking and smart mobility. In *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1735–1739. IEEE, 2013. 112

[59] Markus Hanselmann, Thilo Strauss, Katharina Dormann, and Holger Ulmer. Canet: An unsupervised intrusion detection system for high dimensional can bus data. *Ieee Access*, 8:58194–58205, 2020. 64, 67

[60] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*, pages 784–800, 2018. 130

[61] Jaewoong HEO, Hyunghoon KIM, and Hyo Jin JO. Some/ip intrusion detection system using machine learning. *IEICE TRANSACTIONS on Information and Systems*, 105(11):1923–1924, 2022. 98

[62] Nadine Herold, Stephan-A Posselt, Oliver Hanka, and Georg Carle. Anomaly detection for some/ip using complex event processing. In *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, pages 1221–1226. IEEE, 2016. 79, 80, 82, 84, 97, 99, 101

[63] Chi Mai Kim Ho, Kin-Choong Yow, Zhongwen Zhu, and Sarang Aravamuthan. Network intrusion detection via flow-to-image conversion and vision transformer classification. *IEEE Access*, 10:97780–97793, 2022. 99

[64] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 10, 35

[65] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 35

[66] Md Delwar Hossain, Hiroyuki Inoue, Hideya Ochiai, Doudou Fall, and Youki Kadobayashi. Long short-term memory-based intrusion detection system for in-vehicle controller area network bus. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 10–17. IEEE, 2020. 87

[67] Wenhao Huang, Guojie Song, Man Li, Weisong Hu, and Kunqing Xie. Adaptive weight optimization for classification of imbalanced data. In *International Conference on Intelligent Science and Big Data Engineering*, pages 546–553. Springer, 2013. 83

[68] Marco Iorio, Alberto Buttiglieri, Massimo Reineri, Fulvio Risso, Riccardo Sisto, and Fulvio Valenza. Protecting in-vehicle services: Security-enabled some/ip middleware. *IEEE Vehicular Technology Magazine*, 15(3):77–85, 2020. 80, 97, 98

[69] Marco Iorio, Massimo Reineri, Fulvio Risso, Riccardo Sisto, and Fulvio Valenza. Securing some/ip for in-vehicle service protection. *IEEE Transactions on Vehicular Technology*, 69(11):13450–13466, 2020. 80, 97, 98

[70] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018. 130

[71] Abdul Rehman Javed, Saif Ur Rehman, Mohib Ullah Khan, Mamoun Alazab, and Thippa Reddy. 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. 79

[72] S Jeong, B Jeon, B Chung, and HK Kim. Automotive ethernet intrusion dataset. *IEEE DataPort*, 2021. 113, 116

[73] Seonghoon Jeong, Boosun Jeon, Boheung Chung, and Huy Kang Kim. Convolutional neural network-based intrusion detection system for avtp streams in automotive ethernet-based networks. *Vehicular Communications*, 29:100338, 2021. 80

[74] Seonghoon Jeong, Boosun Jeon, Boheung Chung, and Huy Kang Kim. Convolutional neural network-based intrusion detection system for avtp streams in automotive ethernet-based networks. *Vehicular Communications*, 29:100338, 2021. 109, 113, 115, 116, 117

[75] Tianchu Ji, Shraddhan Jain, Michael Ferdman, Peter Milder, H Andrew Schwartz, and Niranjan Balasubramanian. On the distribution, sparsity, and inference-time quantization of attention values in transformers. *arXiv preprint arXiv:2106.01335*, 2021. 130

[76] Prad Kadambi, Karthikeyan Natesan Ramamurthy, and Visar Berisha. Comparing fisher information regularization with distillation for dnn quantization. 2020. 130

[77] H Kang, B Kwak, YH Lee, H Lee, H Lee, and HK Kim. Car hacking: Attack and defense challenge 2020 dataset. *IEEE Dataport*, 2021. 55

[78] Min-Ju Kang and Je-Won 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. IEEE, 2016. 79

[79] Shah Khalid Khan, Nirajan Shiwakoti, Peter Stasinopoulos, and Yilun Chen. Cyber-attacks in the next-generation cars, mitigation techniques, anticipated readiness and future directions. *Accident Analysis & Prevention*, 148:105837, 2020. 112

[80] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 58

[81] Takuma Koyama, Masashi Tanaka, Asami Miyajima, Shintaro Ukai, Takeshi Sugashima, and Masumi Egawa. Some/ip intrusion detection system using real-time and retroactive anomaly detection. In *2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring)*, pages 1–7. IEEE, 2022. 92

[82] Takuma Koyama, Masashi Tanaka, Asami Miyajima, Shintaro Ukai, Takeshi Sugashima, and Masumi Egawa. Some/ip intrusion detection system using real-time and retroactive anomaly detection. In *2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring)*, pages 1–7. IEEE, 2022. 97, 98

[83] Liangzhen Lai, Naveen Suda, and Vikas Chandra. Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus. *arXiv preprint arXiv:1801.06601*, 2018. 130

[84] Timm Lauser, Daniel Zelle, and Christoph Krauß. Security analysis of automotive protocols. In *Computer Science in Cars Symposium*, pages 1–12, 2020. 80, 97

[85] Van-Hoang Le and Hongyu Zhang. Log-based anomaly detection without log parsing. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 492–504. IEEE, 2021. 50

[86] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 36

[87] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 10

[88] Yukyung Lee, Jina Kim, and Pilsung Kang. Lanobert: System log anomaly detection based on bert masked language model. *arXiv preprint arXiv:2111.09564*, 2021. 50

[89] Gabriel Leen and Donal Heffernan. Expanding automotive electronic systems. *Computer*, 35(1):88–93, 2002. 5

[90] Bai Li, Zining Zhu, Guillaume Thomas, Yang Xu, and Frank Rudzicz. How is bert surprised? layerwise detection of linguistic anomalies. *arXiv preprint arXiv:2105.07452*, 2021. 50

[91] Bai Li, Zining Zhu, Guillaume Thomas, Yang Xu, and Frank Rudzicz. How is bert surprised? layerwise detection of linguistic anomalies. *arXiv preprint arXiv:2105.07452*, 2021. 73

[92] Yi Li, Jing Lin, and Kaiqi Xiong. An adversarial attack defending system for securing in-vehicle networks. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2021. 131

[93] Yuekang Li, Hongxu Chen, Cen Zhang, Siyang Xiong, Chaoyi Liu, and Yi Wang. Ori: A greybox fuzzer for some/ip protocols in automotive ethernet. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*, pages 495–499. IEEE, 2020. 80, 97, 98

[94] Zilong Lin, Yong Shi, and Zhi Xue. Idsgan: Generative adversarial networks for attack generation against intrusion detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 79–91. Springer, 2022. 131

[95] Stefano Longari, Daniel Humberto Nova Valcarcel, Mattia Zago, Michele Carminati, and Stefano 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, 2020. 39

[96] Stefano Longari, Daniel Humberto Nova Valcarcel, Mattia Zago, Michele Carminati, and Stefano 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, 2020. 64

[97] Meng Lu, Kees Wevers, Rob van der Heijden, and Tom Heijer. Adas applications for improving traffic safety. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 4, pages 3995–4002. IEEE, 2004. 112

[98] Hampus Lundberg, Nishat I Mowla, Sarder Fakhrul Abedin, Kyi Thar, Aamir Mahmood, Mikael Gidlund, and Shahid Raza. Experimental analysis of trustworthy in-vehicle intrusion detection system using explainable artificial intelligence (xai). *IEEE Access*, 10:102831–102841, 2022. 131

[99] Bin Ma, Shichun Yang, Zheng Zuo, Bosong Zou, Yaoguang Cao, Xiaoyu Yan, Sida Zhou, and Jichong Li. An authentication and secure communication scheme for in-vehicle networks based on some/ip. *Sensors*, 22(2):647, 2022. 98

[100] Kirsten Matheus and Thomas Königseder. *Automotive ethernet*. Cambridge University Press, 2021. xv, xvii, xix, 5, 6, 20, 22, 26, 78, 81, 115

[101] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. 10

[102] Jörn Migge, Josetxo Villanueva, Nicolas Navet, and Marc Boyer. Insights on the performance and configuration of avb and tsn in automotive ethernet networks. *Proc. Embedded Real-Time Software and Systems (ERTS 2018)*, 2018. 27, 112

[103] Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015(S 91), 2015. 6

[104] Minki Nam, Seungyoung Park, and Duk Soo Kim. Intrusion detection method using bi-directional gpt for in-vehicle controller area networks. *IEEE Access*, 9:124931–124944, 2021. 51

[105] Minki Nam, Seungyoung Park, and Duk Soo Kim. Intrusion detection method using bi-directional gpt for in-vehicle controller area networks. *IEEE Access*, 9:124931–124944, 2021. 99

[106] Harini Narasimhan, R Vinayakumar, and Nazeeruddin Mohammad. Unsupervised deep learning approach for in-vehicle intrusion detection system. *IEEE Consumer Electronics Magazine*, 2021. 64

[107] Sen Nie, Ling Liu, and Yuefeng Du. Free-fall: Hacking tesla from wireless to can bus. *Briefing, Black Hat USA*, 25:1–16, 2017. 6

[108] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021. 37, 39

[109] Francisco Parada-Loira, Elisardo González-Agulla, and José L Alba-Castro. Hand gestures to control infotainment equipment in cars. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 1–6. IEEE, 2014. 112

[110] Seungyoung Park, Myungjin Kim, and Seokwoo Lee. Anomaly detection for http using convolutional autoencoders. *IEEE Access*, 6:70884–70901, 2018. 39

[111] Marek Pawlicki, Michał Choraś, and Rafał Kozik. Defending network intrusion detection systems against adversarial evasion attacks. *Future Generation Computer Systems*, 110:148–154, 2020. 131

[112] Jonathan Petit, Bas Stottelaar, Michael Feiri, and Frank Kargl. Remote attacks on automated vehicles sensors: Experiments on camera and lidar. *Black Hat Europe*, 11(2015):995, 2015. 7

[113] Archana Prabhu, HN Champa, and Deepti Kalasapura. Network intrusion detection using sequence models. In *2019 Grace Hopper Celebration India (GHCI)*, pages 1–5. IEEE, 2019. 87

[114] Manassés Ribeiro, André Eugênio Lazzaretti, and Heitor Silvério Lopes. A study of deep convolutional auto-encoders for anomaly detection in videos. *Pattern Recognition Letters*, 105:13–22, 2018. 39

[115] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 118

[116] Marcel Rumez, Daniel Grimm, Reiner Kriesten, and Eric Sax. An overview of automotive service-oriented architectures and implications for security countermeasures. *IEEE access*, 8:221852–221870, 2020. 80, 97

[117] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. Network anomaly detection using lstm based autoencoder. In *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pages 37–45, 2020. 39

[118] Udo Schneider. Iso/sae 21434-the standard for security in connected cars. *ATZ worldwide*, 124(4):64–64, 2022. 8

[119] MARIA SEALE. Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities. 2022. 130

[120] Eunbi Seo, Hyun Min Song, and Huy Kang Kim. Gids: Gan based intrusion detection system for in-vehicle network. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6. IEEE, 2018. 79

[121] Fahad Siddiqui, Rafiullah Khan, and Sakir Sezer. Bird's-eye view on the automotive cybersecurity landscape & challenges in adopting ai/ml. In *2021 Sixth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 1–6. IEEE, 2021. xv, 7

[122] Hyun Min Song, Jiyoung Woo, and Huy Kang Kim. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, 21:100198, 2020. 79

[123] Mengxuan Tan, Alfonso Iacovazzi, Ngai-Man Man Cheung, and Yuval Elovici. A neural attention model for real-time network intrusion detection. In *2019 IEEE 44th conference on local computer networks (LCN)*, pages 291–299. IEEE, 2019. 99, 104

[124] Ivan Tashev, Mike Seltzer, Yun-Cheng Ju, Ye-Yi Wang, and Alex Acero. Commute ux: Voice enabled in-car infotainment system. In *Mobile HCI'09: Workshop on Speech in Mobile and Pervasive Environments (SiMPE)*, 2009. 112

[125] Adrian Taylor, Sylvain Leblanc, and Nathalie 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. IEEE, 2016. 112

[126] Muhammad Usama, Muhammad Asim, Siddique Latif, Junaid Qadir, et al. Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. In *2019 15th international wireless communications & mobile computing conference (IWCMC)*, pages 78–83. IEEE, 2019. 131

[127] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 10, 16, 40, 50, 52, 53, 65, 92, 97, 102, 103, 104

[128] Miki E Verma, Michael D Iannacone, Robert A Bridges, Samuel C Hollifield, Pablo Moriano, Bill Kay, and Frank L Combs. Addressing the lack of comparability & testing in can intrusion detection research: A comprehensive guide to can ids data & introduction of the road dataset. *arXiv e-prints*, pages arXiv–2012, 2020. 71

[129] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008. 51

[130] Dingwang Wang and Subramaniam Ganesan. Automotive network security. In *2021 IEEE International Conference on Electro Information Technology (EIT)*, pages 193–196. IEEE, 2021. 112

[131] Haomin Wang and Wei Li. Ddostc: A transformer-based network attack detection hybrid mechanism in sdn. *Sensors*, 21(15):5047, 2021. 99

[132] Zheng Wang. Deep learning-based intrusion detection with adversaries. *IEEE Access*, 6:38367–38384, 2018. 131

[133] Arkadiusz Warzyński and Grzegorz Kołaczek. Intrusion detection systems vulnerability on adversarial examples. In *2018 Innovations in Intelligent Systems and Applications (INISTA)*, pages 1–4. IEEE, 2018. 131

[134] Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988. 34

[135] Wufei Wu, Renfa Li, Guoqi Xie, Jiyao An, Yang Bai, Jia Zhou, and Keqin Li. A survey of intrusion detection for in-vehicle networks. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):919–933, 2019. xv, 5, 12

[136] Zihan Wu, Hong Zhang, Penghai Wang, and Zhibo Sun. Rtids: A robust transformer-based approach for intrusion detection system. *IEEE Access*, 10:64375–64387, 2022. 99

[137] Kaichen Yang, Jianqing Liu, Chi Zhang, and Yuguang Fang. Adversarial examples against the deep learning based network intrusion detection systems. In *MILCOM 2018-2018 ieee military communications conference (MILCOM)*, pages 559–564. IEEE, 2018. 131

[138] Yu-Guang Yang, Hong-Mei Fu, Shang Gao, Yi-Hua Zhou, and Wei-Min Shi. Intrusion detection: A model based on the improved vision transformer. *Transactions on Emerging Telecommunications Technologies*, 33(9):e4522, 2022. 99

[139] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access*, 5:21954–21961, 2017. 87

[140] Keping Yu, Liang Tan, Shahid Mumtaz, Saba Al-Rubaye, Anwer Al-Dulaimi, Ali Kashif Bashir, and Farrukh Aslam Khan. Securing critical infrastructures: deep-learning-based threat detection in iiot. *IEEE Communications Magazine*, 59(10):76–82, 2021. 50

[141] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris M Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9813–9823, 2021. 65

[142] Ding Yuxin and Zhu Siyi. Malware detection based on deep learning algorithm. *Neural Computing and Applications*, 31(2):461–472, 2019. 11

[143] Daniel Zelle, Timm Lauser, Dustin Kern, and Christoph Krauß. Analyzing and securing some/ip automotive services with formal and practical methods. In *The 16th International Conference on Availability, Reliability and Security*, pages 1–20, 2021. 96, 98

[144] Daniel Zelle, Timm Lauser, Dustin Kern, and Christoph Krauß. Analyzing and securing some/ip automotive services with formal and practical methods. In *The 16th International Conference on Availability, Reliability and Security*, pages 1–20, 2021. 109

[145] Xing Zhang, Xiaotong Cui, Kefei Cheng, and Liang Zhang. A convolutional encoder network for intrusion detection in controller area networks. In *2020 16th International Conference on Computational Intelligence and Security (CIS)*, pages 366–369. IEEE, 2020. 116

[146] Xiaogang Zhu, Shigang Liu, Xian Li, Sheng Wen, Jun Zhang, Camtepe Seyit, and Yang Xiang. Defuzz: Deep learning guided directed fuzzing. *arXiv preprint arXiv:2010.12149*, 2020. 11

[147] Richard Zuech, Taghi M Khoshgoftaar, and Randall Wald. Intrusion detection and big heterogeneous data: a survey. *Journal of Big Data*, 2(1):1–41, 2015. 61

ECOLE DOCTORALE

Titre : Détection d'intrusion avec les techniques d'apprentissage en profondeur pour les réseaux automobiles embarqués

Mots clés : Détection d'intrusion, Apprentissage en profondeur, Controller Area Network, Automotive Ethernet

Résumé : La communication automobile embarquée, qui désigne la communication et l'échange de données entre les calculateurs embarqués, joue un rôle crucial dans le développement des systèmes de transport intelligents (STI), qui visent à améliorer l'efficacité, la sécurité et la durabilité des systèmes de transport. La prolifération des dispositifs informatiques et de communication embarqués, centrés sur des capteurs connectés à un réseau embarqué (IVN), a permis le développement de fonctions de sécurité et de commodité, notamment la surveillance du véhicule, la réduction du câblage physique et l'amélioration de l'expérience de conduite. Cependant, la complexité et la connectivité croissantes des véhicules modernes suscitent des inquiétudes quant à l'évolution des menaces liées aux réseaux embarqués. Une série de risques de sécurité potentiels peuvent compromettre la sécurité et la fonctionnalité d'un véhicule, mettant en danger la vie des conducteurs et des passagers. De nombreuses approches ont donc été proposées et mises en œuvre pour pallier ce problème, notamment les pare-feu, le cryptage, l'authentification sécurisée et les contrôles d'accès. Comme les mécanismes traditionnels ne parviennent pas à contrer complètement les tentatives d'intrusion, il est nécessaire de mettre en place une contre-mesure défensive complémentaire. Les systèmes de détection d'intrusion (IDS) sont donc considérés comme un élément fondamental de toute infrastructure de sécurité réseau, y compris le RVI. L'objectif principal de cette thèse est d'étudier la capacité des techniques d'apprentissage profond à détecter les intrusions à bord des véhicules. Les algorithmes d'apprentissage profond ont la capacité de traiter de grandes quantités de données et de reconnaître des modèles complexes qui peuvent être difficiles à discerner pour les humains, ce qui les rend bien adaptés à la détection des intrusions dans les IVN. Cependant, comme l'architecture E/E d'un véhicule évolue constamment avec l'apparition de nouvelles technologies et exigences, nous proposons différentes solutions basées sur l'apprentissage profond pour différentes architectures E/E et pour diverses tâches, notamment la détection d'anomalies et la classification.

Title : Intrusion Detection with deep learning for in-vehicle networks

Keywords : Intrusion Detection, Deep learning, Controller Area Networks, Automotive Ethernet

Abstract : In-vehicle communication which refers to the communication and exchange of data between embedded automotive devices plays a crucial role in the development of intelligent transportation systems (ITS), which aim to improve the efficiency, safety, and sustainability of transportation systems. The proliferation of embedded sensor-centric communication and computing devices connected to the in-vehicle network (IVN) has enabled the development of safety and convenience features including vehicle monitoring, physical wiring reduction, and improved driving experience. However, with the increasing complexity and connectivity of modern vehicles, the expanding threat landscape of the IVN is raising concerns. A range of potential security risks can compromise the safety and functionality of a vehicle putting the life of drivers and passengers in danger. Numerous approaches have thus been proposed and implemented to alleviate this issue including firewalls, encryption, and secure authentication and access controls. As traditional mechanisms fail to fully counterattack intrusion attempts, the need for a complementary defensive countermeasure is necessary. Intrusion Detection Systems (IDS) have been thus considered a fundamental component of every network security infrastructure, including IVN. Intrusion detection can be particularly useful in detecting threats that may not be caught by other security measures, such as zero-day vulnerabilities or insider attacks. It can also provide an early warning of a potential attack, allowing car manufacturers to take preventive measures before significant damage occurs. The main objective of this thesis is to investigate the capability of deep learning techniques in detecting in-vehicle intrusions. Deep learning algorithms have the ability to process large amounts of data and recognize complex patterns that may be difficult for humans to discern, making them well-suited for detecting intrusions in IVN. However, since the E/E architecture of a vehicle is constantly evolving as new technologies and requirements emerge, we propose different deep learning-based solutions for different E/E architectures and for various tasks including anomaly detection and classification.

Institut Polytechnique de Paris
91120 Palaiseau, France