NAIST-IS-DD1821419

Doctoral Dissertation

Studies on Deep Learning-based Intrusion Detection Systems for Computer & In-vehicle CAN Bus Networks

Md Delwar Hossain

February 10, 2021

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Md Delwar Hossain

Thesis Committee:

Professor Youki Kadobayashi	(Supervisor)
Professor Keiichi Yasumoto	(Co-supervisor)
Professor Yuichi Hayashi	(Co-supervisor)
Associate Professor Yuzo Taenaka	(Co-supervisor)
Associate Professor Daisuke Miyamoto	(The University of Tokyo)

Studies on Deep Learning-based Intrusion Detection Systems for Computer & In-vehicle CAN Bus Networks*

Md Delwar Hossain

Abstract

The rapid growth of the Internet of Things (IoT) and the ubiquitous nature of the Internet have made life more convenient for human beings. The rise of that social convenience is accompanied by incessant efforts of miscreants to create new tools, techniques, and tactics to destabilize the comfort of the dwellers by attacking computer networks and applications. Even worse, these attacks are being transferred into the increasingly connected cyber-physical systems (CPS), especially the automotive system where the in-vehicle CAN bus network lacks encryption and authentication mechanisms, making them even more vulnerable to some of the attacks that are well-known in traditional computer networks. Additionally, some automotive systems (e.g., the modern car) employ advanced technologies –the Telematics Unit, in-vehicle infotainment (IVI), V2X, etc.—accessible through Bluetooth, Wi-Fi, GPS, etc., thus, augmenting their attack surface. Intrusion Detection Systems are known to be the solution by excellence for detecting and mitigating network attacks, however, based on the recrudescence of attacks, we can affirm that traditional IDSs have failed. Elsewhere, artificial intelligence (AI) or, more specifically, deep learning has shown immense promise in solving lingering issues in other domains: we contend that deep learning can also help make IDSs more efficient.

Hence, in this dissertation, our imperative is to devise new IDS methodologies to protect computer networks and in-vehicle CAN bus networks of automotive systems by leveraging deep learning. First, we thoroughly study the deep

*Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, February 10, 2021.

learning-based IDS for several kinds of critical network attacks such as DoS (Denial of Service), DDoS (Distributed DoS), Brute Force, etc. Subsequently, we investigate how to optimize the deep learning models. Our results illustrate that Long Short-Term Memory (LSTM) can effectively detect network attacks with high accuracy and reasonable detection rates. After ensuring security in computer networks by using deep learning, we transfer our solutions to the automotive systems. Therefore, we propose a deep learning-based IDS for in-vehicle CAN bus networks. Furthermore, for efficiency reasons, we also develop CAN bus network attacks (DoS, Fuzzing, and Spoofing) datasets by using the CAN messages of three distinct car models (Toyota, Subaru, and Suzuki). The results of our experiment demonstrate that our deep learning-based IDS is more effective and robust than existing methodologies.

Keywords:

intrusion detection system, Long Short-Term Memory, convolutional neural network, automotive security, CAN Bus, cybersecurity, deep learning

Contents

List of Figures

List of Tables

1. Introduction

The Internet was introduced to connect the world and it is now becoming more and more ubiquitous. The Internet is a patchwork of technologies, standards, policies, etc., but to keep it simple and in line with our thesis, we define the Internet as a network of computer networks composed of numerous features such as the Internet of Things (IoT). The latter represents the epitome of the ubiquity of the Internet, as IoT has been devised to accelerate digital transformation and make "connecting the world" a reality. So far, billions of devices have been connected to make, among other things, our lives more convenient. The rapid growth of advanced technologies has tremendously impacted human beings' comfort, and those technologies are invading every domain that constitutes society. Recently, we have seen how they are helping cyber-physical systems (CPSs) to reach higher levels of development. Cyber-physical systems are modern engineering systems wherein physical systems are connected via computing networks; physical systems are connected with many sensors, and the internal communication is processed with the embedded physical systems [50, 4]. Amidst several CPSs entities, automotive systems are among the most important entities due to their rapid transformations [50].

Indeed, the car is arguably the most important means of transportation of the modern era. It is said that the modern car's inception dates back to 1886, when Karl Benz introduced a patent for his invention called the Benz Patent-Motorwagen. Since then, the modern car has gone through numerous transformations to become more efficient, reliable and secure.

In regards to enhancing comfort and safety, automotive systems employ many advanced technologies –the Telematics Unit, In-vehicle infotainment (IVI), etc. Generally, users access the aforementioned advanced technologies through the Bluetooth technology, Wi-Fi, and GPS [53]. Regarding maintenance and inspection of the vehicle systems, On-Board Diagnostics (OBD-II) is a commonly used physical diagnostic interface of a modern car. OBD-II was developed in the early 80s with many advanced features to overcome the limitations of the earlier developed OBD and broadly used for the computer-controlled on-board vehicle [72]. There are up to 100 Electronic Control Units (ECU) in the modern automotive system, and the Controller Area Network (CAN) bus system orchestrates

the communication between the ECUs. All the ECUs that reside in the modern vehicle could be monitored through the OBD-II interface. Modern automotive systems employ the Telematics Unit for: connecting the vehicle to the outside world through wireless network, sending/receiving data in several advanced computing systems such as the advanced driver assistance system (ADAS), Vehicle Tracking, Vehicle-to-vehicle communication, vehicle navigation, etc[50].

However, the proliferation of intelligent hacking tools has allowed anyone with basic computer literacy to be able to attack computer networks, that phenomenon is a significant threat to the universal acceptance of digital transformation. Thus, cybersecurity is essential to make safe communication, i.e., facilitate the acceptance of the digital transformation. Cybersecurity is a set of technologies, processes, and operations which has been designed to protect the systems and networks against unauthorized accesses, modifications, and destruction. As we know, network attacks have been observed since the beginning of the Internet and they are still relevant due to numerous attempts of independent hackers, cybercrime organizations and state-sponsored hacking squads to intrude other's computer networks. Even worse, critical network attacks are being transferred into the cyber-physical systems applications, notably into the automotive system where the in-vehicle CAN bus network does not support encryption and authentication mechanisms, rendering it even more vulnerable to some of the attacks that are conventional in traditional computer networks.

1.1 Problem Statement

The world is deeply connected thanks to the rapid evolution of Internet technologies. Simultaneously, we are experiencing the rapid growth of technological weapons of mass destruction that anyone can use with elementary computer literacy. There exist several types of computer network attacks of which intrusion and denial of service (DoS, including distributed DoS (DDoS), Brute Force, BoT) attacks are the most prominent. The aforementioned attacks are of significant concern to the stability of the network systems. Even worse, these attacks are being transferred into the cyber-physical systems (CPSs), especially in automotive systems wherein lack of security mechanism (authentication/encryption) makes them vulnerable to some of the attacks that are familiar in traditional computer networks.

As per CPS's consideration, the modern automobile is a complex piece of technology that employs the Telematics Unit to connect the vehicle to the wireless network for communicating with the outside world. In addition, a technology called In-vehicle infotainment (IVI) is integrated with the modern automobiles for accessing the Internet, TV, etc., and, in general, users get access to the IVI through the Bluetooth technology, Wi-Fi, and GPS. The aforementioned technologies are augmenting the attack surfaces of the modern car. Additionally, limited processing power is also a significant fact. Moreover, a lack of standard security solutions for automotive systems originates considerable attention about safe driving. Furthermore, modern cars are transforming rapidly and integrated with numerous advanced electronic pieces of equipment to make them connected; modern cars are connected to the outside world through the aforementioned external communication interfaces and the latter are indispensable to the modern car's advancement.

Furthermore, the Controller Area Network (CAN) bus system is a central system for managing the communication between the electronic control units (ECUs). Despite its central importance, the CAN bus system doesn't support authentication and authorization mechanisms, i.e., CAN messages are broadcast without basic security features. As a result, an attacker can launch attacks effortlessly into an In-vehicle CAN Bus system. Attackers can compromise the CAN bus system in several ways, including Denial of Service (DoS), Fuzzing and

Spoofing attacks. When an attacker succeeds in compromising the ECUs, they can take control and stop the engine, disable the brakes, turn the lights on/off, etc [43]. Consequently, it proffers the questions of transforming modern cars and safe driving. Henceforth, automotive systems required an effective methodology to detect In-vehicle CAN Bus network attacks regarding safe driving, affirming the digital transformation of automotive systems.

Reflecting these, researchers in academia and industry professionals have been working diligently to detect and mitigate the CAN bus network attacks. However, they proposed several methodologies that appear to be lacking adequate and effective attack detection and mitigation technologies [25, 62]. As a countermeasure, an Intrusion Detection System (IDS) that uses modern techniques, tactics and technologies could be a solution to detect and mitigate the aforementioned critical attacks.

1.2 Research Objectives and Contributions

In this research, our primary objectives and contributions are as follows:

1.2.1 Research Objectives

In this research, our primary aim is to design and develop an effective Intrusion Detection System (IDS) to detect and mitigate critical cyber attacks regarding Computer & In-vehicle CAN bus Network System by applying Deep Learning (DL) approaches.

1. RO1.1: Design and develop an effective DL-based IDS for computer networks.
2. RO1.2: Transfer and adapt RO1.1 into In-vehicle CAN Bus networks

- RO1.1: Design and develop an effective DL-based IDS for computer networks.
  - To develop effective Deep Learning-based Intrusion Detection Systems (IDS) to detect and mitigate critical cyber attacks regarding computer network.
  - To optimize the Deep Learning models to achieve the best detection performance.
  - To investigate different deep learning algorithms based on performance, accuracy, and effectiveness.

RO1.1 - the objective is to optimize deep learning models to achieve reasonable detection accuracy and reduce false positive and false negative rates. We thoroughly investigate how to optimize deep learning models. Our experiment results demonstrate that right hyper-parameter values selection is essential to develop a robust IDS. Additionally, transfer and adapt the above objective into In-vehicle CAN Bus networks. The justification is presented in the later Chapters 3 & 4.

As we affirm, several computer network attacks can transfer into the In-vehicle CAN bus system; hence, we intend to transfer our solution into the In-vehicle CAN Bus system.

- **RO1.2**: Transfer and adapt RO1.1 into In-vehicle CAN Bus networks
  - To develop real systems-based imbalanced attack datasets.
  - To provide an effective pre-processing method for developing an effective supervised classification model.
  - To develop an effective deep learning-based In-vehicle CAN Bus network intrusion detection system (IDS).

RO1.2 - the research's principal aim is to generate CAN Bus network attack datasets from different real-car models by using On-board diagnostics (OBD-II) communication interface and propose an effective pre-processing method to develop a robust IDS for In-vehicle CAN Bus network system. Finally, we investigate different deep learning models to determine the most suitable algorithms for developing the IDS for CAN-bus network attack detection. Although we study datasets from specific car models, we believe our IDS will be effective regarding any models of cars. The aforementioned proposal details are presented later in Chapters 5, 6 & 7.

1.2.2 Research Contributions

1. We study several critical computer network attacks and develop deep learning-based IDS by using LSTM deep learning approach. We achieve reasonable detection rates to classify network attacks such as DoS, DDoS, Brute Force, etc.
   - We thoroughly investigate how to optimize the deep learning approaches to attain higher detection rates.
2. We develop LSTM and 1D CNN deep learning-based intrusion detection systems that are effective in In-vehicle CAN Bus networks despite the pro-

liferation of attacks and the lack of encryption and authentication mechanisms in In-vehicle CAN Bus systems.

3. We develop CAN Bus network attacks (DoS, Fuzzing, Spoofing) datasets by using the CAN Bus messages of a real car.

4. We provide an effective pre-processing method to develop an effective LSTM and 1D CNN-based supervised classification model regarding the CAN bus attack detection.

5. Proposed solutions will provide an effective countermeasures in well-known domains that can be transferred (adapted) to domains that are new playground for hackers.

1.3 Dissertation Outline

The dissertation outline is as follows: In **Chapter 2**, we discuss the preliminaries of Computer Network Attacks and the Controller Area Network (CAN) Bus system, communications interfaces. We also discuss the machine learning models' performance evaluation matrix. **Chapter 3** presents our studies about deep learning-based IDS for SSH and FTP Brute Force attacks. In **Chapter 4**, we explain our proposed LSTM-based IDS regarding several critical network attacks detection. We provide details on how to optimize the model and improve the detection accuracy of the IDS. In **Chapter 5**, we discuss In-vehicle CAN Bus attacks, also detail the development of NAIST CAN attack datasets. In **Chapter 6**, we present our proposed LSTM-based IDS regarding In-vehicle CAN Bus attack detection. **Chapter 7** presents our 1D CNN-based IDS regarding In-vehicle CAN Bus attack detection. In **Chapter 8**, we discuss the future works. **Chapter 9** concludes the dissertation.

2. Preliminaries

This chapter discusses an overview of the CAN bus network and how it works inside the modern car. We further discuss the several kinds of critical Computer & In-vehicle network attacks and attack surfaces and briefly discuss the external communication interfaces for the summary of how to inject attack against the modern car.

2.1 Computer Network Attacks

There exist several types of network attacks of which intrusion, Brute Force and denial of service (DoS, including distributed DoS (DDoS)) attacks are the most prominent.

SSH and FTP brute-force Attack. One of the oldest hacking techniques is the brute-force attack on several services in computer networks such as SSH and FTP. A Brute Force attack can be easily automated, with just minimum attacking knowledge and intervention, attackers can launch brute-force attacks. Several intelligent brute-force attack tools are available: Hydra, Aircrack-ng, John the Ripper, Rainbow Crack, etc. Hydra is one of the most popular brute-force attacking tools, it is available by default in Kali Linux. In a dictionary attack, the attacker uses a word list that contains users' commonly used passwords. Brute Force attacks can be detected at the host and network levels. Researchers have mostly focused on the host-based system to detect Brute Force attacks. For the host-based system, access logs need to be analyzed to detect, for instance, the number of failed login attempts [69]. The network traffic needs to be analyzed to detect network-based Brute Force attacks [51, 70]. We can detect brute-force attacks through an intrusion detection system by using NetFlow data analysis.

DoS/DDoS. DoS and DDoS are arguably the most prominent attacks on computer networks and e-services [29]. DoS/DDoS disrupt the availability of the systems; consequently, they affect the CIA triad. There are mainly two types of DoS/DDoS attacks: high-volume and low-volume. The former is noticeable due to the amount of traffic that is sent to a target network. The latter behaves like a legitimate traffic; thus, making it less noticeable. The variations of low-volume DoS/DDoS attacks are low-rate, slow-rate, and one-shot attacks [21]. For many

decades, academia and the industry have tried to find solutions for DoS/DDoS attacks without much success. However, the recent advancements in machine learning and deep learning have resuscitated the hope that we can find solutions to counter the ever-sophisticated DoS/DDoS attacks in particular, and network attacks in general. There are already trailblazing works in academia [77, 6], and deep learning approaches have particularly been successfully applied to many application domains.

2.2 Controller Area Network (CAN) bus

The modern car has gone through numerous transformations to become more efficient, reliable and secure. The Controller Area Network (CAN) bus protocol is one the most important transformations introduced to the car industry. Developed by Robert Bosch in the 1980s, the CAN is an International Standardization Organization (ISO) defined serial communication bus that is in charge of the flow of information between the Electronic Control Units (ECUs) of a car. In simpler words, the CAN bus coordinates the movements between the engine, the brakes, the steering wheel, etc., i.e., it makes the modern car *connected*. The CAN protocol was initially engineered for industrial machinery, however it has been adopted for vehicular network communications.

The modern car is comprised of about 50 to 100 ECUs, some of which are connected through the CAN bus. The CAN bus protocol is effective for vehicular network systems because of its low cost and centralized system. The ECUs communicate with messages by using the CAN protocol. Each ECU receives messages with unique CAN bus IDs which are used for intra-interactions.

In in-vehicle communication systems, messages are transferred to the vehicle system managed by the CAN bus protocol. Hundreds of sensor data communicate to send messages to the CAN bus system. An ECU can share control data with an outside element of the vehicle through a network system.

Fig. 1 shows the 11 bit mode CAN message format. A standard CAN frame consists of several fields which are Start of Frame (SOF), Arbitration Field (CAN ID), Control Field, Data Field (payload), CRC Field, Acknowledge Field (ACK), and End of Frame (EOF).

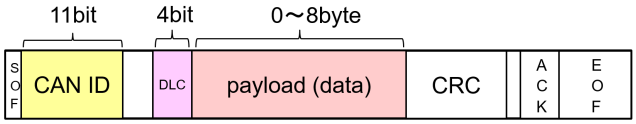


Figure 1. CAN message format in 11bit mode with DLC=8. There are no security features implemented in this protocol.

- **Start of Frame**. The Start of Frame bit is used to synchronize and notify all nodes regarding the start of the CAN messages transmission.
- **CAN ID (Arbitration Field)**. CAN ID is used for an identification number to which ECU the message should be received. The size is 11 bits. The priority of the message is established by this field, in general, a lower value indicates a higher priority.
- **DLC Field**. Data Lenght Code (DLC) is a part of the control field, which indicates the byte length of the Data Field. It ranges between 0 to 8.
- **Data Field (Payload)**. It contains the application payload data, which is interpreted by the received ECUs.
- **CRC Field**. It is used to detect the error regarding the message transmission. CRC field size is 16 bits and it contains the CRC sequence from the SOF to the Data Field.
- **Acknowledge Field**. This filed is used to get the confirmation from the receiver node regarding the proper reception of the CAN message. In case of transmission error detection, the sender can send the CAN message again.
- **End of Frame**. This field indicates the end of the CAN message.

Fig. 2 shows few components of modern automotive systems, which consists of many advanced technologies such as OBD-II, Telematics Unit, In-vehicle Infotainment (IVI), etc.

2.3 Communication Interfaces

An attacker can compromise the In-vehicle CAN bus ECUs by using the External Communication Interfaces such as Telematics Unit, In-vehicle Infotainment (IVI), and On-Board Diagnostics (OBD-II). An attacker tries to inject malicious packets by using the interfaces, as mentioned above. Modern cars are transforming rapidly thanks to the integration of numerous advanced electronic pieces of equipment. Modern cars are connected to the outside world through the aforementioned external communication interfaces [53]. Figure 2 shows the external communication interfaces and a few of the advanced technologies integrated with modern automotive systems. The interfaces mentioned above are indispensable to the modern car's advancement; however, they increase the attack surfaces to compromise the connected car. In this research, we thoroughly investigated how an attacker can compromise the modern car using external communication interfaces.

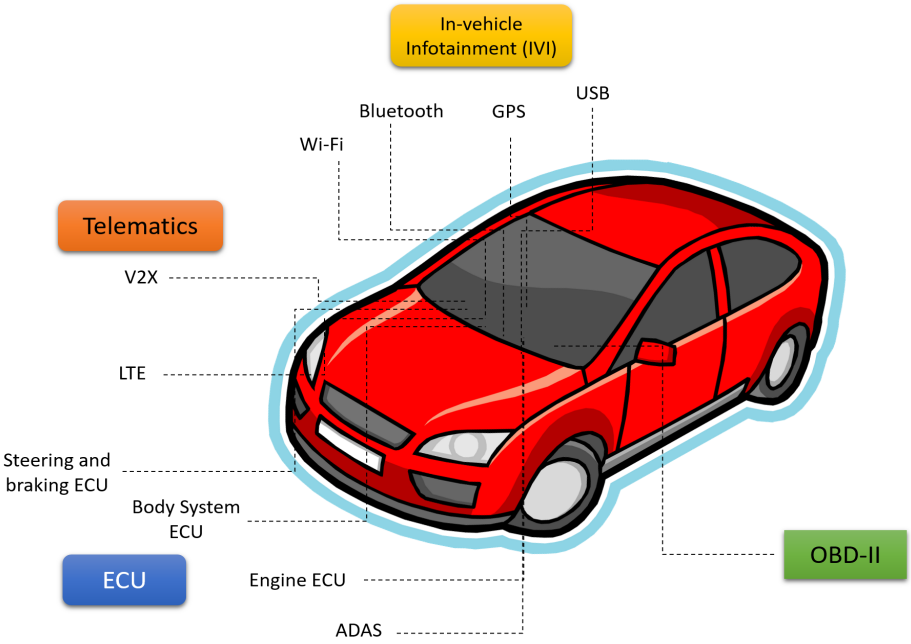


Figure 2. Few Components of Automotive Systems.

2.3.1 On-Board Diagnostics (OBD-II)

On-Board Diagnostics (OBD-II) is one of the most typical physical diagnostic interfaces into the modern car, which is majorly used to extract in-vehicle data concerning vehicle inspection or maintenance. OBD was introduced in the early 80s and broadly employed for the computer-controlled on-board vehicle. OBD-II has developed to modernize the on-board diagnostic features to overwhelm the earlier developed OBD limitations [72]. We can monitor all the ECUs connected to the CAN bus system through the OBD-II port.

In this research, we use the OBD-II port of a real-car to extract data to develop the CAN bus attack dataset. Details regarding how we extract data using OBD-II and develop the attack datasets presented in Chapter 5.

2.3.2 Telematics Unit

The modern vehicle is transforming so rapidly, and numerous electro-mechanical pieces of equipment are embedded into the vehicle system. Telematics units are employed to send/receive data in various advanced computing systems through wireless networks such as the advanced driver assistance system (ADAS), Vehicle Tracking, Vehicle-to-vehicle communication, vehicle navigation, etc. [50].

2.3.3 In-vehicle Infotainment (IVI)

Modern cars are integrated with numerous advanced electro-mechanical pieces of equipment to modernize the connected car, users' entertainment, and safe driving. In-vehicle Infotainment (IVI) is an integrated system to provide information and entertainment for users. IVI can be accessed and make a connection with the external networks through Bluetooth, Wi-Fi, GPS. Diverse advanced automotive instruments are connected with the IVI systems such as V2X, ADAS, Sensors, etc., concerning safe driving and improvement, the user's comfort [50, 57].

2.4 Performance Evaluation Matrix for Machine Learning Model

Performance measurement is an essential aspect in machine learning. We evaluate the network attack detection performance by using the Area Under The Curve (AUC)-Receiver Operating Characteristics (ROC) curves, and F1 scores [3].

The F1 score is an essential factor for measuring machine learning performance evaluation when the datasets are imbalanced. In the case of an imbalanced dataset, we cannot evaluate the performance by detecting the accuracy only. Similar to AUC-ROC, the model is strong if the F1 score is close to 1.0. The F1 score is the weighted average results of precision and recall. We also consider the False Positive Rate (FPR) and False Negative Rate (FNR) regarding the measure of effectiveness of the detection systems.

$$FPR = \frac{FP}{FP + TN} \tag{1}$$

$$FNR = \frac{FN}{FN + TP} \tag{2}$$

Accuracy (Equation 10) is the number of correctly classified attack instances against the total observations [33].

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{3}$$

Where: TP = True Positive; FP = False Positive; TN = True Negative; FN = False Negative.

Recall (Equation 12) is the ratio of correctly predicted positive observations of all the observations in the actual class.

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

The F1 score (Equation 13) is calculated based on the precision and recall [61, 73].

$$F1 - Score = \frac{2 * (Recall * Precision)}{Recall + Precision} \tag{5}$$

2.5 Chapter Summary

This chapter discusses the preliminaries of several kinds of critical network attacks. This chapter also discusses the automotive communication interfaces and a few advanced technologies/equipment regarding the modern car related to our research. We provide an overview of a few advanced pieces of automotive systems equipment and how an attacker can compromise automotive systems using external communication interfaces.

3. Studies on LSTM-based Intrusion Detection System

3.1 Introduction

Network traffic anomaly detection is of critical importance in cybersecurity due to the massive and rapid growth of sophisticated computer network attacks. Indeed, the more new Internet-related technologies are created, the more elaborate the attacks become. Among all the contemporary high-level attacks, dictionary-based brute-force attacks (BFA) present one of the most unsurmountable challenges. There are numerous security measures – firewalls, antivirus, intrusion detection systems (IDS) [9, 16, 20, 54, 60, 68] – that have been elaborated throughout the years to detect and protect entire network systems. The IDS provides vital contributions thanks to its early alarm system against the network attacks. Aforementioned, attacks can significantly disrupt the cyber-physical systems (CPS) network systems. For instance, CPS are connected by means of the Internet of Things (IoT) devices; hence, they are susceptible to brute-force attacks which are prominent in IoT. We need to develop effective methods to detect and mitigate such brute-force attacks in real-time.

Throughout the years, academia and the industry have worked together to develop the different genre of IDS based on the functionalities and requirements: signature-, and anomaly-based [39, 71]. Malicious activities can be monitored in the network with effective and high performing intrusion detection systems (IDS). Deep packet inspection (DPI)-based traditional IDS encounter difficulties to be implemented in encrypted and high-speed networks. In contrast, network flow-based IDS are effective in the aforementioned settings. Indeed, flow-based IDS rely on network flow data instead of the payload information of the packets.

Brute-force attacks (BFA) [69] on SSH and FTP can be detected by a host and network-based approaches. In case of host-based detection, we need to install the software at the individual host; wherein network-based detection, which we contemplate in this research, we can identify malicious activities such as BFA by just monitoring the network. An effective and high performing method is imperative to detect dictionary-based SSH and FTP brute-force attacks at the

network level. Because of the vast amount of data communications, most of them being, it is obligatory to properly analyze the network traffic flow data to distinguish malicious activities.

In this chapter, we focus on network-level brute-force attack detection on the SSH and FTP protocols by applying and analyzing the LSTM deep learning algorithm on network flow data. We also compare the effectiveness LSTM and Multilayer Perceptron (MLP) feedforward artificial neural network models.

3.2 Related Work

There are numerous academic papers where researchers propose solutions to mitigate BFA attacks. Despite the big number of academic outputs, BFA are still prevalent and, particularly, there is not yet an efficient method for thwarting SSH and FTP attacks. Nevertheless, in the following, we summarize some of the best work on the aforementioned attacks to provide an overview of the importance of our work.

In paper [70], the authors investigated the SSH dictionary attack pattern by analyzing the NetFlow data under the decision tree model. They evaluated their proposal in a high-speed university network. Stiawan et al. explored a time-sensitive statistical relationship approach and visualization of the brute-force attack patterns in an IoT network environment. [64]. They successfully detected the brute-force attack. Najafabadi et al. experimented with SSH brute-force attack detection at the network-level by using NetFlow data analysis [51]. They generated a labeled attack dataset for attack detection by using machine learning approaches, which have proven to be successful in detecting brute-force attacks. In paper [10], the authors analyzed the network intrusion detection system by leveraging LSTM. They used the CIDDS-001 dataset and they achieved an accuracy score of 0.85. Kim et al. delved into a neural network classifier for intrusion detection by using LSTM along with the KDDCUP99 dataset [37]. They compared the results of the LSTM approach with machine learning algorithms results, and the former outperformed the latter. In paper [31], the authors studied distributed SSH brute-force attacks and investigated an eight-year dataset of thousands of SSH users. They demonstrated that some of the individual attack detection methods are somewhat difficult to implement. Satoh et al. [56] analyzed

Table 1. CICIDS2017 - Benign and Attack Instances

Dataset	Attack	Number of Instances
Tuesday	Benign	432074
	FTP-Patator	7938
	SSH-Patator	5897
Thursday	Benign	168186
	Web Attack-brute-force	1507
	Web Attack-XSS	652
	Web Attack-SQL Injection	21

flow analysis-based SSH dictionary attack detection, and they afterward proposed a method developed by two novel elements. In paper [26], the authors studied SSH intrusion detection. They concluded that it is difficult to detect the attack at a high-speed network with the conventional approach of Deep Packet Inspection. In paper [18], authors considered protocol-independent dictionary attacks detection.

In the research we surveyed, researchers studied brute-force attacks detection by using different methodologies. We use LSTM and NetFlow data analysis to detect FTP and SSH brute-force attacks.

3.3 CICIDS2017 Dataset

In this research, we consider the CICIDS2017 [59] dataset to detect the SSH and FTP brute-force attacks. The CICIDS2017 dataset consists of several kinds of network attacks; the dataset contains five days of attack dataset; each day consists of different types of network attacks. We consider the Tuesday and Thursday dataset, which consists of brute-force attacks. We discussed the SSH and FTP brute-force attack scenarios in Section 2.1.

3.3.1 Dataset CICIDS2017 (Tuesday)

CICIDS2017 [59] is a widely used dataset for attack detection. The dataset contains common attacks and benign data. There are five days of attacks and normal data in this dataset. The attack dataset was developed with the Patator brute-

Table 2. LSTM Parameter Values

Parameters	**Value**
Activation Function Input	tanh
Output Layer	4
Epoch	200
Activation Function Output	softmax
Optimizer	RMSprop
Dropout	0.1
Batch Size	512
Loss Function	categorical_crossentropy
Encoder	Label Encoder

force tool, in a controlled environment which allowed to include the most up to date attacks. Our aim is to investigate SSH and FTP related brute-force attacks. We consider the Tuesday, July 4, 2017 sub-dataset which contains benign, FTP, and SSH brute-force labeled attacks. There are 79 features in the CSV file. Table 1 shows that the number of benign elements is higher than the number of attack elements.

3.3.2 Dataset CICIDS2017 (Thursday)

We also consider another day (Thursday, July 6, 2017) of data in the CICIDS2017 dataset. There are three types of labeled attacks in this sub-dataset: web attack brute-force (WABF), web attack XSS (WAXSS), and web attack SQL injection (WASQL). The sub-dataset contains 79 features and is labeled as per attack type. The PCAP data, with full packets payload, is also publicly available for researchers. Table 1 describes the sub-dataset in details.

3.4 LSTM-based Brute-force Attack Detection

Long Short Term Memory networks (LSTMs) is a special kind of Recurrent Neural Networks (RNN). LSTM was introduced by Hochreiter and Hochreiter in 1997 [27]. In this research, we experiment with a Stacked LSTM model [12, 13] with four hidden layers and we apply multiclass classification. Table 2 provides the

LSTM parameter details which we use for our experiment. We use the four LSTM stack layers on top of each other with arbitrary neuron settings: 512-256-128-32, and the final output is a dense layer with *softmax* activation, which gets input from the last layer output of the LSTM layer.

In this LSTM model experiment, we use the *RMSprop* optimizer with a learning rate of 0.0001, and we let the remaining parameters to their default values. We employ *categorical_crossentropy* as loss function because of the multiclass classification. We split the dataset into training and testing sets wherein 80% of the data is used for training and 20% is used for testing. The validation dataset was specified to the *fit()* function.

We use python PyCharm IDE 2019.2.2 and Keras with TensorFlow as backend. All the experiments were carried out on Intel Core i7 CPU 2.20 GHz, 16 GB RAM, Windows 10 (64-bit), NVIDIA GeForce GTX 1050. As per Table 2, we apply *categorical_crossentropy* as loss function, *RMSprop* optimizer and *softmax* as an activation function output for SSH and FTP brute-force attack detection. We use *categorical_crossentropy* as loss function, *RMSprop* optimizer and *softmax* as activation function outputs regarding the *Wednesday* dataset.

Due to the fact that benign class elements are high in the dataset, the detection accuracy with the proposed deep learning classifiers is also high. So, we considered individual attack detection by using precision, recall, and F1-score, and we use area under the curve (AUC) and receiver operating characteristics (ROC) curves to evaluate the performance. We investigate SSH and FTP brute-force attack detection performance and the classifiers' classification accuracy.

3.4.1 Applying LSTM to CICIDS2017 (Tuesday)

In the last decade (2010 - 2019), deep learning has been successfully applied in many fields such as image recognition, data processing, voice recognition, etc. Thus, it is only normal to think that deep learning can also be successfully leveraged for cyber attacks detection. The CICIDS2017 (Tuesday) dataset contains 79 features and labels composed of benign and attack elements. We split the dataset as follows: 80% for training, and 20% for testing. As a reminder, the dataset contains benign elements, FTP and SSH attacks and the number of benign instances are higher than the number of FTP and SSH attack instances.

In this experiment, we use the *RMSprop* optimizer. We set the optimizer learning rate to 0.0001 and we let the remaining parameter of the optimizers to their default values. We use *categorical_crossentropy* as loss function. We choose 200 epochs as the number of iterations. We set the validation data by using the *fit()* function.

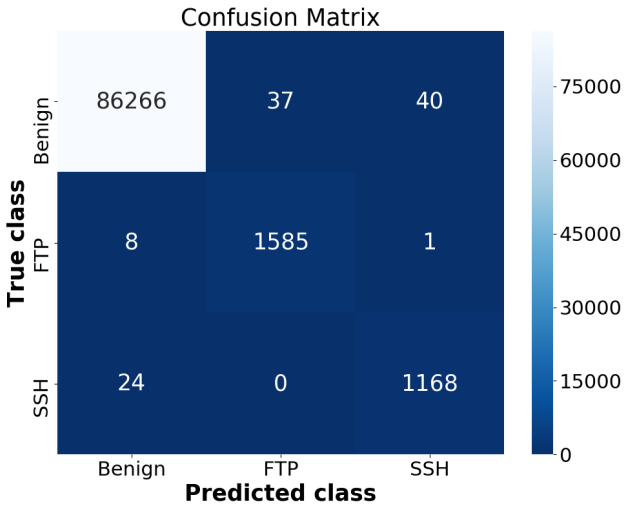


Figure 3. LSTM Confusion Matrix - FTP and SSH Attacks

We train the model with benign and attacks classes. The proposed model can classify the attack classes with high accuracy and low false positive (FP) and false-negative detection rates. Fig. 3 and Table 4 show the LSTM confusion matrix (CM) for the *Tuesday* dataset and Table 5 provides the classification results for both datasets. The CM demonstrates that our model effectively classified the FTP and SSH brute-force attacks with low false positive and false negative rates, which are 0.0039 and 0.0088, respectively. The accuracy of the LSTM model for FTP and SSH brute-force attacks is 99.88%. In Table 3, the results of individual attacks detection for FTP and SSH are respectively as follows: precision, 0.98 and 0.97; recall, 0.99 and 0.98; F1-score, 0.99 and 0.97. These results are additional proofs that our LSTM model is effective in detecting FTP and SSH brute-force attacks at the network level.

Table 4 describes the confusion matrix regarding FTP and SSH attacks detection based on the LSTM and MLP models. As per the confusion matrix, LSTM

Table 3. LSTM Classification Results - FTP and SSH Attacks

	Precision	**Recall**	**F1-Score**	**Support**
Benign	1.00	1.00	1.00	86343
FTP-Patator	0.98	0.99	0.99	1594
SSH-Patator	0.97	0.98	0.97	1192
Accuracy			1.00	89129
Macro avg	0.98	0.99	0.99	89129
Weighted avg	1.00	1.00	1.00	89129

Table 4. Confusion Matrix - FTP and SSH Attacks

Method	**Datasets**	**Benign**	**FTP**	**SSH**
LSTM	Benign	86266	37	40
	FTP	8	1585	1
	SSH	24	0	1168
MLP	Benign	86007	85	251
	FTP	358	1234	2
	SSH	19	1	1172

is effectively detecting the FTP and SSH brute-force attacks. The MLP model has a lower attack detection rate than the LSTM model.

3.4.2 Applying LSTM to CICIDS2017 (Thursday)

In this section, we discuss the experiment results regarding the CICIDS2017 (Thursday) dataset. The dataset contains benign, web attack brute-force (WABF), web attack XSS (WAXSS), and web attack SQL (WASQL) instances. We applied our LSTM model and analyzed the performance of overall and individual attack detection. Fig. 4 depicts the LSTM confusion matrix for the *Thursday* dataset. Table 2 provides the parameters of the model. The accuracy of our LSTM model is 98.86%. As per Table 6, the precision of the web attack-brute-force detection is 0.94, the recall is 0.11, and the F1-score is 0.19.

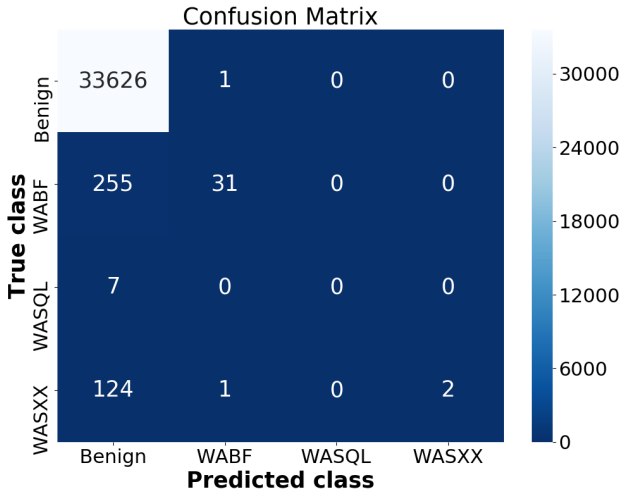


Figure 4. LSTM Confusion Matrix - CICIDS2017 (Thursday)

Table 5. Classification Results LSTM vs MLP

Dataset	Method	Accuracy	TPR	TNR	FPR	FNR
Tuesday	LSTM	99.88%	0.9912	0.9961	0.0039	0.0088
	MLP	99.20%	0.8897	0.9380	0.0620	0.1103
Thursday	LSTM	98.86%	0.2790	0.7708	0.2292	0.7210
	MLP	98.81%	0.2762	0.7690	0.2310	0.7238

3.4.3 Discussion

From the evaluation results, we could see that LSTM achieves reasonable detection accuracy. However, this is simply because the benign instances are higher than the instances of SSH and FTP attacks in the dataset. That is why we considered precision, recall, F1-score in the performance analysis. In the confusion matrix analysis, we observed that overall weighted classifier accuracy, precision, recall, and F1-score were high. F1-score needs to be higher when it comes to implementing such kinds of models in real-time environments. Our proposed LSTM-based IDS is effective in detecting the FTP and SSH brute-force attacks at the network system. We observed that LSTM did not perform well for the detection of web brute-force attacks, web attack XSS, and web attack SQL injection.

Table 6. LSTM Classification Results of CICIDS2017 (Thursday)

	Precision	**Recall**	**F1-Score**	**Support**
Benign	0.99	1.00	0.99	33627
WA-brute-force	0.94	0.11	0.19	286
WA-SQL	0.00	0.00	0.00	7
WA-XSS	1.00	0.02	0.03	127
Accuracy			0.99	34047
Macro avg	0.73	0.28	0.30	34047
Weighted avg	0.99	0.99	0.98	34047

MLP was also unable to achieve a reasonable detection rate.

Although Brute Force is one of the old and common attacks, it is still prominent and critical for the network system. In this research, we consider a supervised classification model and labeled dataset. In the future, we will consider developing our own attack dataset to evaluate the detailed attack pattern and behavior of the network before and after the Brute Force attack injection. We will further consider an unsupervised method to detect Brute Force attacks. Furthermore, we will consider how to reduce false positive and negative rates.

3.5 Chapter Summary

In this chapter, we studied an LSTM-based IDS for SSH and FTP brute-force attacks. We trained the model in a dataset containing benign and brute-force attack class instances. Our model can predict anomalous network traffic behavior and identify malicious activities in network systems. Our experimental results demonstrated that the proposed LSTM model could classify the SSH and FTP brute-force attacks with high accuracy of 99.88% while having low false positive and false negative rates. We investigated SSH and FTP brute-force attack detection with LSTM and MLP. We compared the performance of attack detection among those methods. These results indicate that we need to develop more robust IDS in detecting critical network attacks and reducing the false positive and false negative rates.

4. Optimization of Deep Learning Models by Hyper-parameter Values Tuning

4.1 Introduction

Cybersecurity is the cornerstone of the Internet. It is a set of technologies, processes, and operations, which has been designed to protect the computer systems' confidentiality, integrity, and availability (CIA). Among all of its many components, network traffic attack detection is known to be one the most critical as it allows to prevent and/or detect potentially destructive attacks. Moreover, network attacks have been around since the beginning of the Internet and they are still relevant due to the numerous attempts of independent hackers, cybercrime organizations and state-sponsored hacking squads to intrude into others computer networks. Simultaneously, the rapid growth of the attacker's skills and the proliferation of intelligent hacking tools exacerbate the situation by increasing the attack surface on computer networks. In response, firewalls, antivirus, intrusion detection systems (IDS) [60, 9, 54, 36] have been devised to protect the computer networks and systems. There exist several types of network attacks of which intrusion and denial of service (DoS, including distributed DoS (DDoS)) attacks are the most prominent [29]. We discuss the DoS and DDoS attacks in Section 2.1.

In this chapter, we leverage the artificial recurrent neural network architecture Long Short-Term Memory (LSTM) to detect network-level attacks such as slow-rate DoS, DDoS LOIT, BoT ARES, and port scan. Furthermore, our aim is to optimize the model to improve the network attack detection rate, reduce false positives by fine-tuning different LSTM hyper-parameters: optimizers, loss functions, learning rates and activation functions, and by comparing their performance. We evaluate the performance considering detection accuracy, F1 score, and AUC-ROC curve area because the number of attack elements are lower than the number of benign elements. We evaluate the performance of individual attacks detection by using precision, recall, and F1-score. We use a multiclass classification model for our experiment and we consider overall and individual attack detection performance. We investigate the optimum hyper-parameter values

by fine-tuning LSTM parameters such as layers, optimizers, activation functions, learning rates, loss functions, and by comparing the different performance.

4.2 Related Work

We can argue that network attack detection research is as old as the Internet; thus, there are thousands of academic papers in this field. In this section, we discuss network attack detection systems that use deep learning models.

In paper [9], the authors analyzed the network intrusion detection system by using LSTM. They applied their methodology on the CIDDS-001 dataset and achieved an accuracy score of 0.85. Le et al. investigated LSTM-based IDS by using the KDD CUP99 dataset [36]. They experimented and compared the model's performance with six optimizers. Their results show that the *Nadam* optimizer provided the best performance. Jihyun et al. proposed neural network classifiers for intrusion detection by leveraging LSTM-RNN in the KDDCUP99 dataset [38]. They also compared their results with different machine learning algorithms. LSTM-RNN achieved an accuracy score of 96.93. Gao et al. worked on an intrusion detection system that takes advantage of Deep Belief Network (DBN) [23]. They achieved an accuracy of 93.49% while working with the KDD CUP 1999 dataset. Alrawashdeh et al. used Restricted Boltzmann Machine (RBM) and DBN to develop an intrusion detection system, which they applied to the KDD-CUP'99 dataset for their experiment [8]. They achieved a detection rate of 97.9%. Zahangir Alom et al. investigated another intrusion detection system based on DBN [7]. They used the NSL-KDD dataset for their investigation. The proposed model achieved a 97.5% accuracy score. Zhao et al. engineered a DBN and Probabilistic Neural Network (PNN)-based intrusion detection system [80], and made use of the KDDCUP'99 dataset. Their solution optimized DBN-PNN achieved a detection rate of 93.25%. Radford et al. explored network traffic anomaly detection based on LSTM model [52]. They used the CICIDS2017 dataset for their experiment. In paper [77], the authors analyzed deep learning-based DDoS attack detection, and they also compared their LSTM model's performance with the random forest (RF) machine learning classifier. They achieved an accuracy score of 97.606% and 93.627% by LSTM and RF, respectively. Alkasassbeh et al. delved into DDoS attack detection by using Multilayer Perceptron (MLP), Naïve

Bayes, and Random Forest; MLP achieved an accuracy score of 98.63% [6].

It is essential to have a recent dataset comprised of the most up-to-date attacks in order to devise efficient and effective network attack detection systems. Most of the researchers use rather old datasets to test their proposals, which does not make sense and there is a growing concern in the community for not using datasets that are contemporaneous to the proposed solution. Effective detection accuracy rate, i.e., the reduction of the false positive alarm is the key factor for real-time network attack detection systems. In deep learning models, to improve the detection accuracy, it is important to select the best parameter values. In this research, we employ the CICIDS2017 [59] dataset for our exploration and we investigate several hyper-parameter value changes to find the best parameter values for detecting network attacks.

4.3 Dataset CICIDS2017

4.3.1 Dataset CICIDS2017 (Wednesday)

We use the CICIDS2017 labeled dataset for our experiment. The CICIDS2017 dataset is widely used for network attack detection [59]. The most common network attacks are available in the dataset. CICIDS2017 contains five days of attacks and benign data. Our goal is to investigate DoS, DDoS, port scan and BoT attacks. Hence, we consider the *Wednesday*, July 5, 2017 dataset, which contains several kinds of DoS labeled attacks, including slow-rate DoS attacks, SlowHTTPTest and Slowloris. In the *Friday*, July 7, 2017 data, they produced separate attack datasets. The *Morning* dataset contains BoT ARES, the *Afternoon* dataset is comprised of separate datasets of port scan and DDoS attacks. We concatenate all three datasets of the *Friday* data into one and we applied multiclass classification. We drop the NaN and Infinity values from the dataset in our experiment.

4.3.2 Dataset CICIDS2017 (Friday)

The *Friday* dataset contains DDoS LOIT, BoT ARES, and port scan attacks, with 79 features. Table 7 provides the attack and benign instances, and the number of instances in the *Wednesday* and *Friday* datasets. The *Wednesday DoS* dataset

Table 7. CICIDS2017 - Benign and Attack Instances

Dataset	Attack	Number of Instances
Wednesday	Benign	440031
	DoS GoldenEye	10293
	DoS Hulk	231073
	DoS SlowHTTPTest	5499
	DoS Slowloris	5796
	Heartbleed	11
Friday	Benign	414322
	BoT ARES	1966
	DDoS LOIT	128027
	port scan	158930

contains 692703 elements wherein DoS SlowHTTPTest, Slowloris and Heartbleed are 5499, 5796, and 11 elements, respectively, which are lower compared to other attack instances. Those attacks are challenging to detect effectively with high accuracy. We concatenate all three datasets of the Friday data into one. There are 703245 attack and benign instances wherein 1966 instances are BoT ARES, 128027 are DDoS and 158930 are port scan.

4.4 LSTM-based Network Attack Detection

In our experiment, we used python PyCharm IDE 2019.2.2 and Keras [2] with TensorFlow as backend. All the experiments were performed on an Intel Core i7 laptop, with a CPU of 2.20 GHz, a RAM of 16 GB, Windows 10 (64-bit) as operating system, and NVIDIA GeForce GTX 1050. We applied *categorical_cross entropy* as loss function, the *RMSprop* optimizer was used with the default values of the parameter and *softmax* was used as activation function output.

4.4.1 Applying the Long Short-Term Memory (LSTM) Model

We experiment with a Stacked LSTM model with five hidden layers and arbitrary units settings: 512-512-256-128-64, and we apply multiclass classification. Table

8 provides the LSTM parameter details which we use for our experiment. We use five LSTM stack layers on top of each other with arbitrary neuron settings. The final result is a dense layer with *softmax* activation which gets input from the last layer output, the LSTM layer.

We experiment with changes in the parameter values, the number of layers, activation functions, learning rates and loss functions and we use single to six layers (L1-L6) to evaluate the best performance for network attack detection. We use the parameter values described in Table 8 and as per the experimental result, LSTM layer 5 can classify most of the attack class elements with reasonable detection accuracy- hence, we consider layer 5 (L5) for further experiments.

We experiment hyper-parameter values tuning in layer 5 with the parameters described in Table 8, and with the following arbitrary units 512-512-256-128-64. We discuss the details regarding the hyper-parameter values changing experiment settings and results in Section V.

In this LSTM model experiment, we use *RMSprop* optimizer and compare the performance with other optimizers. We set the learning rate to 0.0001, we let the remaining parameters to their default values as per the settings of the optimizer. We use *categorical_crossentropy* as loss function. We split the dataset into training and testing sets wherein 80% of the data was used for training and the remaining 20% was used for testing. The validation dataset was specified to the *fit()* function by the use of validation data. We compare the performance by hyper-parameter tuning. We observed that Layer 5 provides high accuracy and a reasonable detection rate regarding DoS SlowHTTPTest, Slowloris, and Heartbleed attacks.

4.5 Experiment Results and Performance Evaluation

In the following section, we discuss the experimental results, the effectiveness and performance of the model based on hyper-parameter values tuning. The performance evaluation matrix is available in section 2.6.

Table 8. LSTM Model Hyper-parameter Values

Hyper-parameter name	**Value**
Activation Function Input	tanh
Epoch	300
Activation Function Output	softmax
Optimizer	RMSprop
Learning Rate	0.0001
Batch Size	512
Loss Function	categorical_crossentropy
Encoder	Label Encoder

4.5.1 LSTM Layer-wise Attacks Classification - Experiment Results

We experiment with all the attacks by 1-6 LSTM layers. Table 8 shows the parameter values which we applied to our experiment; we compared the performance among the different layers. As per Table 9 and Fig. 7, Layers 5 and 6 provide better detection rate regarding individual DoS attacks detection; the precision, recall, and F1 score are also high. Layers 5-6 can detect DoS Slowloris, SlowHTTPTest and Heartbleed attacks with a reasonable score. L5 detection accuracy is 99.08% wherein detection rate regarding DoS SlowHTTPTest, Slowloris, and Heartbleed attacks are 0.88, 0.76 and 1.00, respectively. We observe that L5-L6 classified the Heartbleed attack with a detection rate of 1.00. The Wednesday dataset contains 692703 elements wherein only 11 elements are heartbleed attacks and our classifier correctly classified the heartbleed attack with a 1.00 detection rate. Fig. 5 depicts the LSTM L5 confusion matrix where we observe that Heartbleed attacks are classified without false negative and false positive. Regarding DoS Goldeneye, Hulk, Slowloris, SlowHTTPTest have some false positive and false negative elements.

As per Table 10, we observe that L5-L6 provide the higher detection accuracy, which are 99.54% and 99.49%, compare to L1-L4 layers. L5 DDoS LOIT and port scan detection rate are 0.9960 and 0.9984, respectively. We observe that the detection rate regarding BoT ARES is lower through L1-L6 layers.

We investigated with hyper-parameter tuning by using layer 5 and evaluated

the performance. We observed that *RMSprop* optimizer with a learning rate of 0.0001 provides a better model performance in the training and testing sets. Regarding DDoS LOIT, BoT ARES and port scan, L5 and L6 have approximately similar detection accuracy, and they outperform the remaining layers. Fig. 6 depicts the LSTM L5 confusion matrix. The confusion matrix shows that our model can reasonably detect most of the DDoS LOIT and port scan attacks with low false positive and false negative rate wherein BoT false positive and false negative rate are higher.

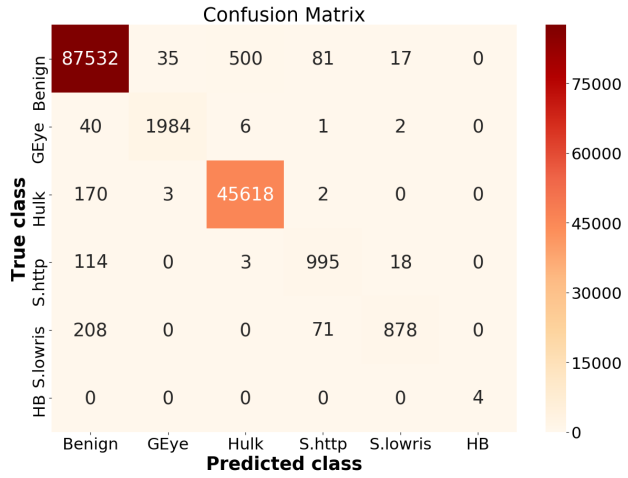


Figure 5. LSTM L5 Confusion Matrix - Wednesday Dataset

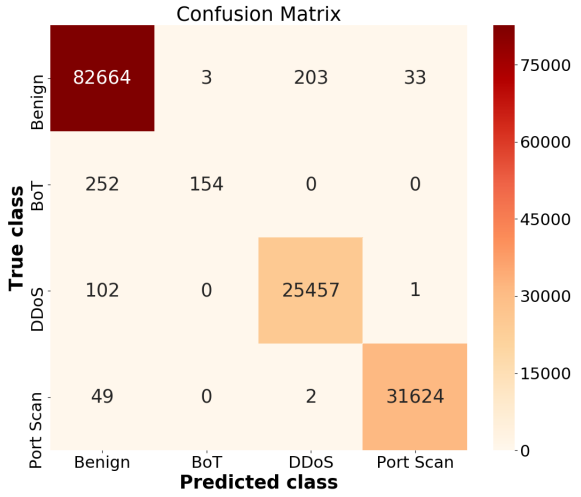


Figure 6. Layer 5 Confusion Matrix - Friday Dataset

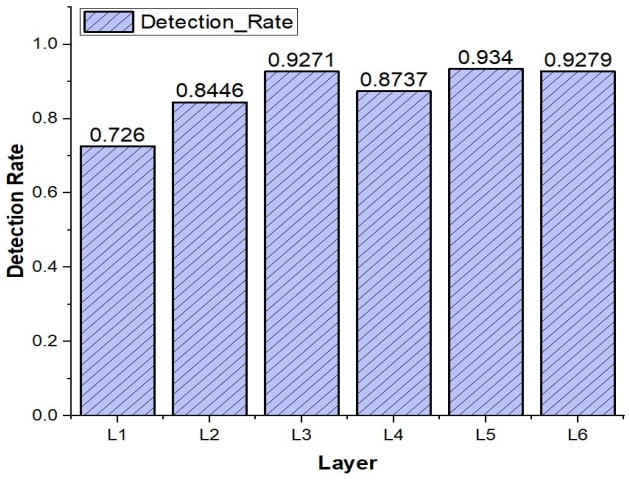


Figure 7. Layer-wise Attacks Detection Rate - Wednesday Dataset

4.5.2 RMSprop Learning Rate wise LSTM Classification Results

Gradient descent is widely used, and it is one of the most popular optimizer algorithms for optimizing the neural networks [55]. The learning rate is one of the key factors regarding gradient descent and other optimizers as well [22]. The

Table 9. Layer-wise Attacks Classification Results - Wednesday Dataset

Layer	Attack	Acc	Recall	F1	FPR	FNR
	Benign		0.9848	0.9862	0.0216	0.0152
	GEye		0.8800	0.9042	0.0010	0.1200
	Hulk		0.9902	0.9823	0.0128	0.0098
L1	Slowhttp	98.11%	0.8097	0.8437	0.0009	0.1903
	Slowloris		0.6914	0.7816	0.0007	0.3086
	H.bleed		0.0000	0.0000	0.0000	1.0000
	Average		0.7260	0.7497	0.0062	0.2740
	Benign		0.9900	0.9919	0.0108	0.0100
	GEye		0.9547	0.9623	0.0004	0.0453
	Hulk		0.9973	0.9905	0.0081	0.0027
L2	Slowhttp	98.89%	0.8761	0.8784	0.0010	0.1239
	Slowloris		0.7494	0.8373	0.0003	0.2506
	H.bleed		0.5000	0.6667	0.0000	0.5000
	Average		0.8446	0.8879	0.0034	0.1554
	Benign		0.9904	0.9915	0.0130	0.0096
	GEye		0.9597	0.9694	0.0003	0.0403
	Hulk		0.9959	0.9897	0.0082	0.0041
L3	Slowhttp	98.87%	0.8372	0.8833	0.0005	0.1628
	Slowloris		0.7796	0.8611	0.0003	0.2204
	H.bleed		1.0000	1.0000	0.0000	0.0000
	Average		0.9271	0.9492	0.0037	0.0729
	Benign		0.9901	0.9914	0.0129	0.0099
	GEye		0.9710	0.9729	0.0004	0.0290
	Hulk		0.9977	0.9906	0.0082	0.0023
L4	Slowhttp	98.86%	0.8212	0.8637	0.0007	0.1788
	Slowloris		0.7122	0.8224	0.0002	0.2878
	H.bleed		0.7500	0.8571	0.0000	0.2500
	Average		0.8737	0.9163	0.0037	0.1263
	Benign		0.9928	0.9934	0.0106	0.0072
	GEye		0.9759	0.9785	0.0003	0.0241
	Hulk		**0.9962**	0.9926	0.0055	0.0038
L5	Slowhttp	**99.08%**	**0.8805**	0.8728	0.0011	0.1195
	Slowloris		**0.7589**	0.8475	0.0003	0.2411
	H.bleed		**1.0000**	1.0000	0.0000	0.0000
	Average		**0.9340**	0.9475	0.0030	0.0660
	Benign		0.9905	0.9889	0.0224	0.0095
	GEye		0.9710	0.9751	0.0003	0.0290
	Hulk		0.9844	0.9844	0.0077	0.0156
L6	Slowhttp	98.52%	**0.8575**	0.8679	0.0010	0.1425
	Slowloris		**0.7640**	0.8545	0.0002	0.2360
	H.bleed		**1.0000**	1.0000	0.0000	0.0000
	Average		0.9279	0.9451	0.0053	0.0721

usage of a large value for the learning rate will cause radical changes to the values of the weights and bias, i.e., we may overpass the global minima. It is difficult to

Table 10. Layer-wise Attacks Classification Results - Friday Dataset

Layer	Attack	Acc	Recall	F1	FPR	FNR
L1	Benign	98.90%	0.9913	0.9908	0.0141	0.0087
	BoT ARES		0.3128	0.4686	0.0001	0.6872
	DDoS LOIT		0.9829	0.9789	0.0056	0.0171
	port scan		0.9967	0.9972	0.0007	0.0033
	Average		0.8209	0.8589	0.0051	0.1791
L2	Benign	99.24%	0.9958	0.9936	0.0124	0.0042
	BoT ARES		0.3670	0.5284	0.0001	0.6330
	DDoS LOIT		0.9844	0.9868	0.0024	0.0156
	port scan		0.9981	0.9980	0.0006	0.0019
	Average		0.8363	0.8767	0.0039	0.1637
L3	Benign	99.40%	0.9965	0.9950	0.0094	0.0035
	BoT ARES		0.3695	0.5357	0.0000	0.6305
	DDoS LOIT		0.9908	0.9901	0.0023	0.0092
	port scan		0.9980	0.9985	0.0003	0.0020
	Average		0.8387	0.8798	0.0030	0.1613
L4	Benign	99.41%	0.9956	0.9950	0.0080	0.0044
	BoT ARES		0.3867	0.5558	0.0000	0.6133
	DDoS LOIT		0.9937	0.9899	0.0031	0.0063
	port scan		0.9981	0.9989	0.0001	0.0019
	Average		0.8435	0.8849	0.0028	0.1565
L5	Benign	**99.54%**	0.9971	0.9961	0.0070	0.0029
	BoT ARES		**0.3793**	0.5471	0.0000	0.6207
	DDoS LOIT		**0.9960**	0.9940	0.0018	0.0040
	port scan		**0.9984**	0.9987	0.0003	0.0016
	Average		**0.8427**	0.8840	0.0023	0.1573
L6	Benign	**99.49%**	0.9971	0.9957	0.0080	0.0029
	BoT ARES		**0.4015**	0.5621	0.0001	0.5985
	DDoS LOIT		**0.9932**	0.9929	0.0017	0.0068
	port scan		**0.9984**	0.9985	0.0004	0.0016
	Average		0.8475	0.8873	0.0026	0.1525

converge to the global minima by using a large learning rate. We can reduce the risk regarding overpassing the minima by using a lower learning rate instead of a large learning rate. The limitation behind using the smaller learning rate is that it will increase the time to converge; thus, the training period is longer [22].

The *RMSprop* optimizer was introduced by Geoff Hinton in the coursera lecture class [67]. *RMSprop* optimizer's default learning rate is 0.001. It is suggested to use default parameter values of the *RMSprop* optimizer except for the learning rate, we can freely tune the learning rate. We use the *RMSprop* optimizer for our experiment to detect DoS, DDoS, BoT and port scan attacks. We evaluated the performance with *RMSprop* optimizer and a learning rate of 0.0001. We showed in Section 5.A that LSTM layer 5 provides a reasonable attack detection accuracy. Hence, we investigate the performance of Layer 5 by changing the learning rate of the optimizer and letting the remaining optimizers to their default settings. Figs. 8 and 9 show the experimental results based on the different learning rates. We observe that a 0.0001 learning rate provides the best detection accuracy, the best recall and a better F1 score in addition to low FPR and FNR to individual attacks detection when compared to other learning rates [17]. Our experiment results show how the learning rate affects the accuracy. A small learning rate provides high accuracy, it is the opposite for a large learning rate.

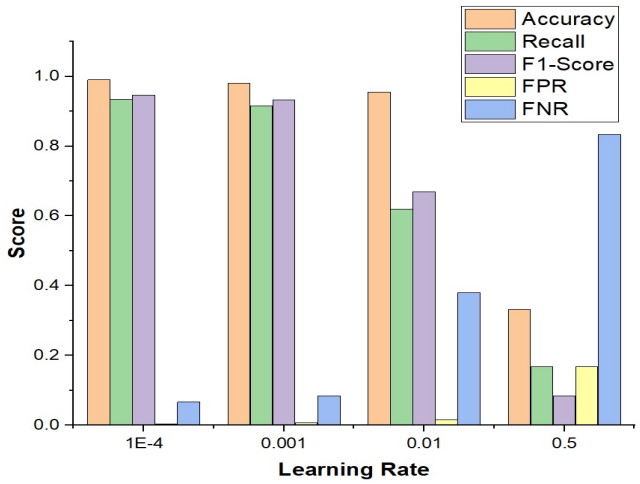


Figure 8. Learning Rate-wise Classification Results - Wednesday Dataset

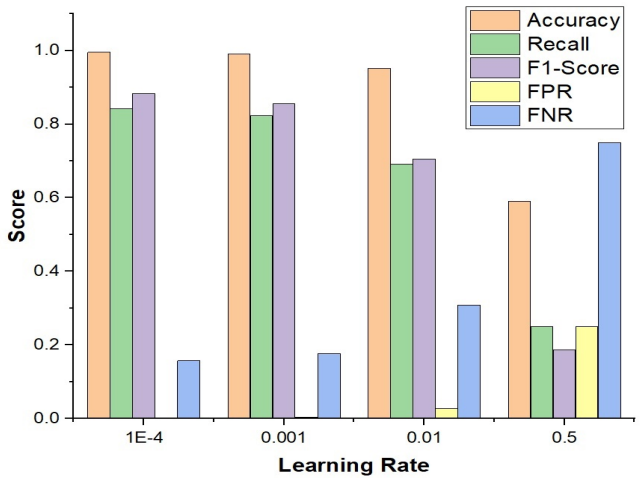


Figure 9. Learning Rate-wise Classification Results - Friday Dataset

4.5.3 Optimizer-wise LSTM Classification Results

Gradient descent is one of the most commonly used optimization algorithms to optimize the deep neural network model in terms of performance optimization and in-general. Three different gradient descent models are available: batch gradient descent, stochastic gradient descent, and mini-batch gradient descent. Based on the amount of data in the dataset, to compute the gradient of the objective function, we have to select the right gradient descent model [55].

For our investigation of network attack detection we use the *RMSprop* optimizer with a learning rate of 0.0001 and we compare the results with LSTM Layer 5, the settings of the experiment are described in Table 8. A suitable optimizer is essential to get a better detection accuracy that is why we conducted our experiment with the other optimizers in their default settings. We ran experiment with six optimizers: *RMSprop*, *Adam*, *Adagrad*, *Adadelta*, *Adamax* and *Nadam*. Table 11 and Fig. 10 show that *RMSprop* and *Adam* provide high detection rate of 0.9340 and 0.9231, respectively for the Wednesday dataset and *RMSprop* and *Adamax* provide detection rate of 0.8427 and 0.8448, respectively for the Friday dataset. We observe that *Adadelta* and *Adagrad* provide the lowest detection rate regarding the attacks elements classification wherein *RMSprop* is a suitable optimizer for network attack detection, it performs well compared to other optimizers [19, 78, 40].

Table 11. Optimizer-wise Classification Results

Optimizer	DS	Acc	P	R	F1	AUC
RMSprop		**99.08%**	**0.9648**	**0.9340**	**0.9475**	**1.00**
Adam		98.54%	0.9544	**0.9231**	0.9362	1.00
Adagrad	W.day	63.76%	0.1063	0.1667	0.1298	0.71
Adadelta		63.76%	0.1063	0.1667	0.1298	0.71
Adamax		**98.96%**	0.9627	0.7953	0.8400	1.00
Nadam		98.00%	0.9189	**0.9206**	0.9154	1.00
RMSprop		**99.54%**	**0.9917**	**0.8427**	**0.8840**	**1.00**
Adam		98.69%	0.9854	0.8215	0.8575	1.00
Adagrad	Friday	66.05%	0.3913	0.3527	0.3385	0.82
Adadelta		58.99%	0.1475	0.2500	0.1855	0.81
Adamax		**99.52%**	**0.9889**	**0.8448**	**0.8859**	**1.00**
Nadam		98.98%	0.9894	0.8401	0.8798	1.00

DS-Dataset, Acc - Accuracy, P - Precision, R -Recall.

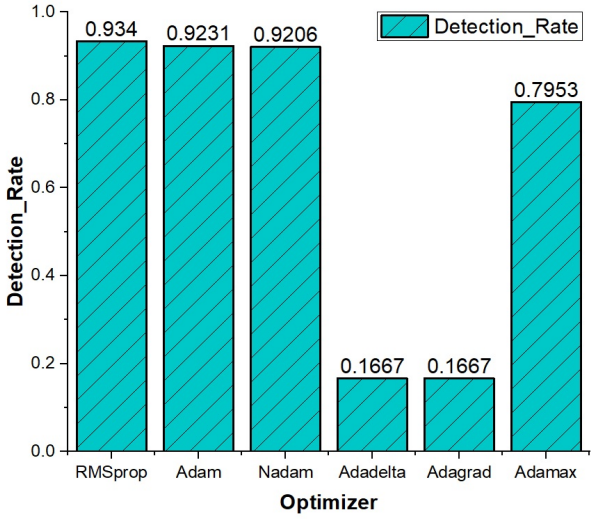


Figure 10. Optimizer-wise Attacks Detection Rate - Wednesday Dataset

4.5.4 Activation Function-wise LSTM Classification Results

We use the *tanh* as an input activation function and *softmax* as an output activation function for attacks detection with the LSTM model's parameter settings

detailed in Table 8. We investigated LSTM layer 5 by only changing the input activation function to *relu* and *sigmoid*, and the output activation function *softmax* remained the same. Table 12 shows the activation functions experiment results. As per the experiment results, *tanh* and *sigmoid* input activation functions provide the highest accuracy scores which are 99.08% and 98.92% for the Wednesday dataset and 99.54% and 99.43% for the Friday dataset.

Table 12. Activation Function-wise Classification Results

Activation Function	**DS**	**Acc**	**P**	**R**	**F1 Score**	**AUC -ROC**
tanh	W.day	**99.08%**	**0.9648**	**0.9340**	**0.9475**	**1.00**
relu		95.02%	0.1588	0.3278	0.3223	0.65
sigmoid		98.92%	0.7995	0.7633	0.7803	1.00
tanh	Friday	**99.54%**	**0.9917**	**0.8427**	**0.8840**	**1.00**
relu		99.10%	0.9891	0.8225	0.8628	1.00
sigmoid		**99.43%**	0.9729	0.8485	0.8871	1.00

4.5.5 Loss Function-wise LSTM Classification Results

To predict the result accurately and optimize the model and reduce the prediction error, We have to select the appropriate loss function for the model. There are a few factors to consider before choosing the appropriate loss function. We have to choose the appropriate loss function based on the particular predictive modeling problem, such as classification or regression [11].

Loss function is one of the key parameters which is essential to get a better network attack detection accuracy. We investigated four loss functions: *categorical_crossentropy*, *mean_absolute_error*, *mean_squared_error*, *kullback_leibler_divergence* with layer 5 settings. Table 13, we observed that *categorical_crossentropy* loss function performs well and provides high detection accuracy. As per our experimental results we observed that *categorical_crossentropy* provides a reasonable detection accuracy and AUC-ROC score regarding DoS attacks detection and both *categorical_crossentropy* and *kullback_leibler_divergence* provide similar accuracy regarding BoT, DDoS and port scan attacks detection. Table 13 provides

the comparison results wherein the *kullback_leibler_divergence* classification accuracy are 98.87% and 99.52% for the Wednesday and Friday datasets, respectively.

Table 13. Loss Function-wise Classification Results

Loss Function	**DS**	**Acc**	**P**	**R**	**F1 Score**	**AUC**
categorical _crossentropy	W.day	**99.08%**	**0.9648**	**0.9340**	**0.9475**	**1.00**
MAE		88.86%	0.3053	0.2946	0.2973	0.54
MSE		98.85%	0.8016	0.7605	0.7788	0.85
KL_divergence		**98.87%**	0.8113	0.7539	0.7796	0.93
categorical _crossentropy	Friday	**99.54%**	**0.9917**	**0.8427**	**0.8840**	**1.00**
MAE		94.64%	0.7213	0.6860	0.7004	0.89
MSE		99.18%	0.9743	0.8412	0.8793	1.00
KL_divergence		**99.52%**	0.9970	0.8399	0.8825	1.00

Acc-Accuracy, DS - Dataset, Acc - Accuracy, P - Precision, R -Recall.

4.6 Discussion

We need an effective network attacks detection system with a high detection accuracy to mitigate the network attacks and stabilize the computer networks and e-services. Several kinds of DoS attacks, especially slow-rate DoS attacks such as SlowHTTPTest, Slowloris, etc., are challenging to detect effectively. We achieved a reasonable attack detection score by applying our LSTM deep learning model. A publicly available labeled dataset is one of the key factors to evaluate the performance regarding network attack detection systems. We investigated the CICIDS2017 labeled dataset which contains several kinds of DoS attacks including the slow-rate DoS attacks. We also examined DDoS LOIT, BoT ARES, and Heartbleed attacks, and we achieved reasonable detection accuracy, precision, recall and F1 score. However, BoT ARES detection precision was high but recall and F1 score were low.

We carefully exploited hyper-parameter tuning. We show how the detection

accuracy varied by hyper-parameter value changes. As per our experimental results, we provided detailed results and we found the best hyper-parameter values for developing a robust intrusion detection system. Our experimental results demonstrate how hyper-parameter values affect the accuracy. In gradient descent, the learning rate is one of the essential factors, it is better to use a smaller learning rate instead of large learning rate to achieve reasonable detection rate. The experiment results demonstrated that deep learning models are effective for network attack detection systems. Our model classified the network attacks with high detection accuracy, precision, recall, F1 score, and AUC-ROC curves. The experiment results proved that the LSTM model is effective in detecting the network attacks.

4.7 Chapter Summary

In this chapter, we studied with a labeled public dataset using a supervised method. Our proposed LSTM model can classify most of the critical attacks with a reasonable detection rate. Furthermore, we investigated how to optimize the deep learning model. Our hyper-parameter-wise experiment results demonstrate that it is necessary to select the right hyper-parameter values to develop an effective IDS. The results indicate, deep learning-based IDS can outperform in the detection network attacks. Our investigation regarding network attack detection with hyper-parameter changes is to provide better directions for developing robust network attacks detection by employing deep learning models.

The experiment results revealed that the LSTM detection accuracy was high, F1 score and AUC-ROC were also high, i.e., the proposed model is efficient for network attack detection. LSTM can effectively classify most of the attacks but we observe that, for BoT ARES detection, the precision, the recall, and the F1 score are low. Our model could detect the DoS GoldenEye, Hulk, and Heartbleed with high precision, recall and F1 score wherein Slowloris and SlowHTTPTest detection rates were 0.76 and 0.88 respectively. As per the hyper-parameter values tuning experiment, we observed that the *RMSprop* optimizer provides better classification results for multiclass classification problems. We compared the performance for the layers 1 through 6, Layer 5 provided the best detection accuracy with 99.08% regarding DoS/DDoS attacks detection, BoT ARES and

port scan detection accuracy was 99.54%. We noticed that *tanh* and *sigmoid* input activation functions provided high detection accuracy compared to *relu*.

5. Develop NAIST CAN Attack Dataset

5.1 Attacks Used in the Model

We mainly experiment with three types of attacks in this research: DoS, Fuzzing, and Spoofing.

5.1.1 Attack Types

Several CAN bus network attacks such as DoS, Fuzzing, and Spoofing are considered the most prominent attacks due to consideration of the damage capability. The modern vehicle includes many ECUs that utilize to communicate with internal and external devices, which increase the attack surfaces. We explain a summary of the DoS, Fuzing, and Spoofing attacks to provide an overview to clarify and understand the attack pattern.

DoS Attack. Flooding is one of the critical threats regarding the CAN bus network system and, thus, safe driving of the car. In case an attacker compromises a car, they can easily inject flooding attacks into the CAN bus network which can cause the interruption or disabling of the services in the ECUs of automobiles. Regarding flooding attacks, attackers continuously send arbitrary messages with high priority bits. Thus, high frequency and high priority CAN messages are transmitted into the CAN bus network system, and the system is occupied by these messages. As a result, the attacks interrupt the transmission of legitimate messages into the CAN bus network system and disable some functions of the ECUs.

Fuzzing Attack. Despite the absence of proper information on the CAN messages, a malicious user can easily attack an in-vehicle network with a Fuzzing attack. During a Fuzzing attack, an attacker injects random ID, DLC, and data fields into the CAN bus, which mimic the legitimate traffic, in the CAN bus system. The Fuzzing attack's disruptions of the CAN bus system manifest as follows: shaking of the steering wheel, signal lights turning on/off erratically, the gear shift changing automatically, etc., [43]. Attackers aim to compromise the ECUs by randomly injecting arbitrary messages. A vehicle controlled by an attacker may be identified by comparing the original CAN ID to its actual CAN ID.

Spoofing Attack. Regarding the Spoofing attack, an attacker injects the modified CAN message considering a specific CAN ID. In case the CAN IDs of the attackers are incorporated in the real system CAN ID, then the ECUs get biased. Consequently, it is difficult to identify legitimate messages, the system may start to malfunction.

5.2 Develop NAIST CAN Attack dataset

5.2.1 Dataset Generation - NAIST CAN attack dataset

We design three attack scenarios –DoS, Fuzzing, and Spoofing– by creating the attack datasets based on the real car messages. Fig. 13 depicts an example of the NAIST CAN attack dataset. The dataset features contain Timestamp, CAN ID, DLC, Payload Data [D0-D7], and Label column. CAN ID is an identifier of the CAN messages. DLC data bytes range from 0-8. The data field contains 64 bits in maximum, and we position each byte in specific columns such as D0-D7. The label represents the original data or the attack data which are added for an experiment.

5.2.2 Data Collection Setup and Process

In this experiment, we collect the CAN message from an actual Toyota hybrid car. We capture the attack-free messages using a CAN analysis tool named Vehicle Spy 3 [65] and the OBD-II interface for a duration of 120 seconds. In this experiment, we develop two datasets. To reiterate, we collected the attack-free data from a real car as a first dataset: Toyota and we obtained the second dataset by injecting DoS, Fuzzing, Speed, and RPM Spoofing attacks. Fig. 11 and Fig. 12 depict the attack injection scenarios with the injections timing wherein the vast amount of attack messages are injected into the CAN within a short amount of time. Regarding the Spoofing attack, injected messages are lower in the same period of the original traffic.

5.2.3 Attack Scenarios

CAN messages are broadcast in the in-vehicle network system; there are no adequate security measures regarding the CAN messages. In fact, there are lackings

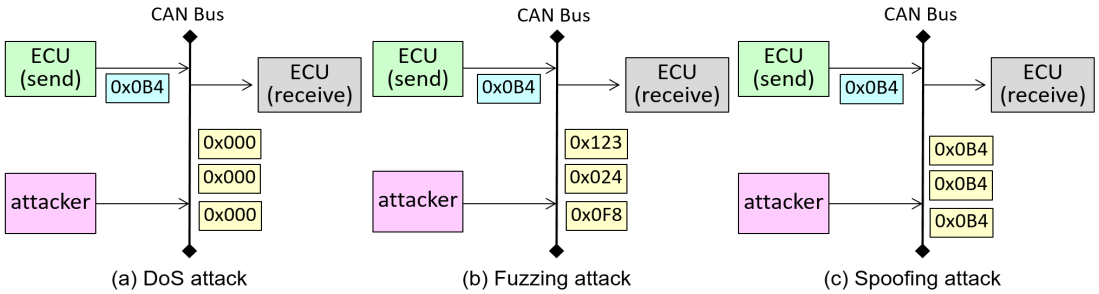


Figure 11. Attack scenarios assumed in this research. (a) DoS attack – an attacker floods messages to the CAN bus. (b) Fuzzing attack – an attacker injects random CAN messages for changing the IDs, payload length. (c) Spoofing attack – an attacker generates fake messages that deceive the receiver's ECUs.

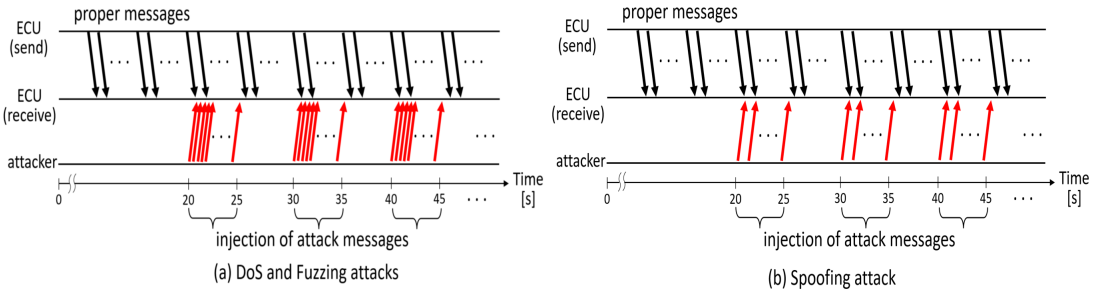


Figure 12. Injection of Messages with Timing Regarding DoS, Fuzzing and Spoofing attacks.

of authentication and encryption mechanisms into the CAN bus system. As a result, it is easy for attackers to disrupt the CAN bus system's confidentiality, integrity, and availability. We consider three critical attacks: DoS, Fuzzing, and Spoofing. The aforementioned attacks are considered critical because they can render the CAN system useless. We develop two datasets in the experiment for each scenario for comparison, one in which we collect data from a real car without any attack data; we obtain the second dataset by injecting attacks.

With the existing algorithm we can develop a single attack at a time. Regarding the DoS attack, we use 0.5 ms as the interval time. The DoS attack interval time can be between 0.1 ms and 1.0 ms. The DoS attack interval time 0.5ms

Table 14. NAIST CAN Attack Dataset - Benign and Attack Instances

Type of Attacks	**Number of Instances**
Benign	947931 (69.22%)
DoS	286502 (20.92%)
Fuzzing	114027 (8.33%)
Spoofing	21072 (1.54%)

produces less amount of DoS attack elements. We use the 1.0 ms interval time to generate the Fuzzing attack; this interval time allows us to inject the maximum amount of Fuzzing attack elements to make the CAN bus system malfunction. We develop the Spoofing attack for a 5-second period and a 5-second duration to produce a smaller amount of Spoofing attack classes (1.54%) compared to the benign classes. We try to mimic real-life scenarios to analyze the imbalance impacts in our experiments and evaluate the performance of the IDS.

DoS Attack

During a DoS attack, the CAN bus system is flooded with messages; thus, the ECUs' regular communications trigger an interruption and the CAN network becomes unavailable to legitimate users. We develop the DoS attack dataset by injecting a large number of messages with the CAN ID 00, DLC 8, data 00. In our experiment, an attack typically starts from 20 seconds for a 10-second period and the interval time is 0.5 ms and the attack duration is 5 seconds. Fig. 13(a) depicts the example of the NAIST DoS attack dataset.

Fuzzing Attack

Regarding the Fuzzing attack, an attacker randomly injects a vast amount of CAN messages with arbitrary data. We use random CAN IDs between 0x000 and 0x7FF with arbitrary lengths. The attack starts from 20 seconds and lasts 5 seconds before resuming after another 5 seconds, and so on and so forth, the interval time is 1ms. Fig. 13(b) depicts the example of the NAIST Fuzzing attack dataset.

Spoofing Attack We inject *handle angel* and *vehicle speed* Spoofing attacks into the Spoofing attack dataset. We inject Spoofing for 10 seconds, for a 5-second duration and 5-second rest(no attack), and the attack starts from 20 seconds. Regarding *handle angle* Spoofing, we use CAN ID 025, DLC 8, data

xx, yy, 00, 02, 5F, FE, 00, CS. The interval time is 12 ms. Regarding *vehicle speed* spoofing, we use CAN ID 0B4, DLC = 8, data 00, 00, 00, CT, 11, xx, yy, CS. Where xx and yy are spoofed values, CS is a checksum, and CT is a counter value. We consider a 25 ms interval for speed Spoofing attack.

Timestamp	ID	DLC	D0	D1	D2	D3	D4	D5	D6	D7	Label
19.990101	127	8	0	10	0	8	1C	33	6E	5	Benign
19.990640	260	8	0	0	0	0	0	0	0	6A	Benign
19.990888	3A0	8	0	0	0	0	0	0	0	52	Benign
19.995562	20	4	0	0	7	2B	-1	-1	-1	-1	Benign
19.995790	230	7	0	0	0	0	0	0	39	-1	Benign
19.996032	25	8	0	4C	0	2	D0	74	0	BF	Benign
19.996630	24	8	2	4	2	0B	42	8	80	9	Benign
20.000433	0	8	0	0	0	0	0	0	0	0	DoS
20.000933	0	8	0	0	0	0	0	0	0	0	DoS
20.001433	0	8	0	0	0	0	0	0	0	0	DoS
20.001933	0	8	0	0	0	0	0	0	0	0	DoS
20.002433	0	8	0	0	0	0	0	0	0	0	DoS
20.002933	0	8	0	0	0	0	0	0	0	0	DoS
20.003481	245	5	0	0	43	20	AF	-1	-1	-1	Benign
20.003981	0	8	0	0	0	0	0	0	0	0	DoS
20.004525	AA	8	1E	EE	1E	2D	1E	B2	1D	E4	Benign

(a) DoS Dataset

Timestamp	ID	DLC	D0	D1	D2	D3	D4	D5	D6	D7	Label
19.992775	320	8	0	0	0	0	1E	0	0	49	Benign
19.997541	1C4	8	3	75	0	0	0	0	0	45	Benign
19.998477	20	4	0	0	7	2B	-1	-1	-1	-1	Benign
19.998719	B4	8	0	0	0	0	90	4	2E	7E	Benign
19.999533	24	8	2	5E	2	1D	42	1B	80	88	Benign
20.000000	23C	8	40	C7	1A	49	7E	43	D6	EF	Fuzzing
20.001000	3C	8	6F	72	D9	19	36	9E	49	84	Fuzzing
20.002000	CB	8	D8	30	E9	BE	42	FD	34	0D	Fuzzing
20.003481	245	5	0	0	43	20	AF	-1	-1	-1	Benign
20.004000	387	8	D8	4C	0A	57	E7	88	DF	A1	Fuzzing
20.004525	AA	8	1E	EE	1E	2D	1E	B2	1D	E4	Benign
20.004764	127	8	0	10	0	8	58	32	6E	40	Benign
20.005014	224	8	0	0	0	0	0	0	0	8	Benign

(b) Fuzzing Dataset

Figure 13. Examples of NAIST CAN Attack Dataset

5.3 Develop NAIST In-vehicle CAN attack dataset

5.3.1 Dataset Generation - NAIST In-vehicle CAN attack dataset

In Section 5.2, we develop a NAIST CAN attack dataset considering a single model of car. We extended and developed the CAN Bus attack dataset for our experiment, consisting of three different car models: Toyota, Subaru, and Suzuki, consisting of DoS, Fuzzing and Spoofing (RPM and Gear) attacks. Fig. 16 depicts an example of the NAIST In-vehicle CAN attack dataset.

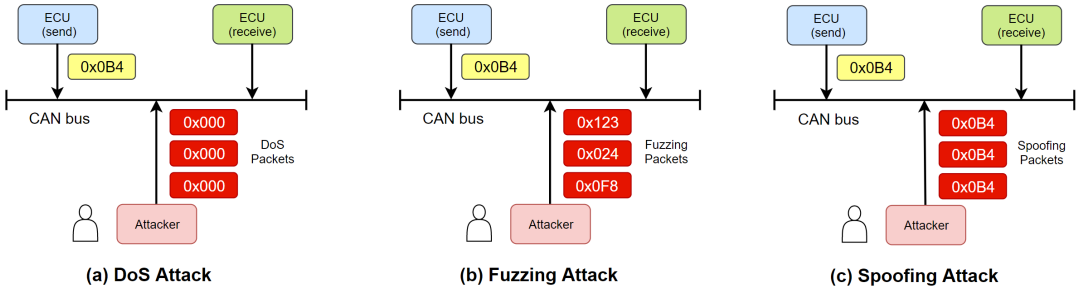


Figure 14. Attack Scenarios

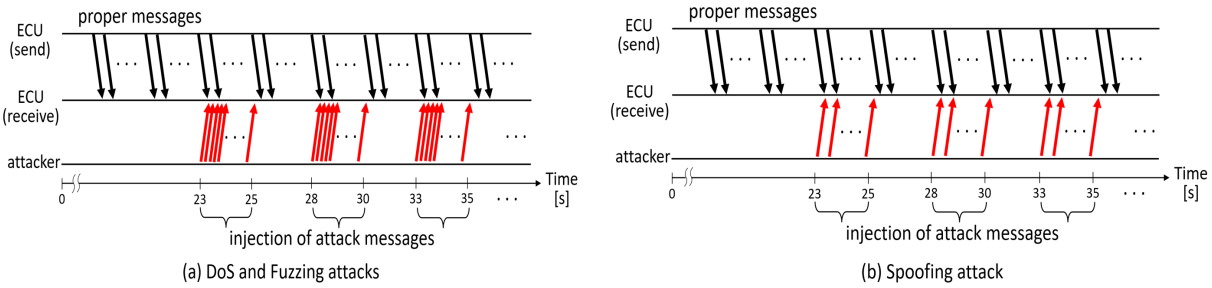


Figure 15. Injection of Messages

5.3.2 Data Collection Setup and Process

We captured the attack-free traffic from the aforementioned real cars with a period of 120 seconds by using CAN analyzers such as the Vehicle Spy 3 Professional and the OBD-II interface. For our experiment, we develop two datasets. To reiterate, we collected the attack-free data from three different models of real cars as

a first dataset: Toyota, Subaru, and Suzuki, and we obtained the second dataset by injecting DoS, Fuzzing, RPM, and Gear Spoofing attacks. Fig. 14 and Fig. 15 depict the attack injection scenarios with the injections timing wherein the vast amount of attack messages are injected into the CAN within a short amount of time. Regarding the Spoofing attack, injected messages are lower in the same period of the original traffic.

5.3.3 Attack Scenarios

CAN messages are broadcast in the in-vehicle network system with no adequate security measures. Indeed, there are no authentication and encryption mechanisms that are available for CAN messages in standard protocol. As a result, it is easy for attackers to violate the confidentiality, integrity, and availability of the CAN bus system. We consider three critical attacks: DoS, Fuzzing, and Spoofing. Such kinds of attacks are considered crucial because they can render the CAN system useless.

Regarding the DoS attack, the interval time is 0.3 ms. DoS attack interval time can be between 0.1 ms and 1.0 ms. We developed a little slower – 0.3ms – DoS attack to produce less amount of DoS attack elements. We developed the fuzzing attack with a 1 ms interval. It will produce a vast amount of fuzzing attacks and will induce malfunction to the CAN bus system. We developed the spoofing attack for a 5-seconds period and a 2-seconds duration because it will produce a smaller amount of attack classes (0.15% 0.40%) compared to the benign classes. The dataset is obviously imbalanced. We try to mimic real life scenarios where normal traffic dominates anomalous traffic in order to analyze how the imbalance impacts our experiments.

DoS Attack. As per the CAN messages characteristics, those CAN messages contain high priority bits that must be accepted first. During a Flooding attack, flood messages contain high priority bits that are flooded into the CAN bus system; thus, they can cause a malfunction to the entire ECUs communication and an interruption of the CAN bus network system. The DoS dataset was developed by injecting a large number of messages with the CAN ID 00, DLC 8 and data 00. In our scenario, the attack injection was for a 5-seconds period and a 2-seconds duration, and the interval time was 0.5 ms.

Table 15. NAIST In-vehicle CAN attack Datasets - Benign and Attack Instances

Attack	**Toyota**	**Subaru**	**Suzuki**
Benign	413752 (74.73%)	710524 (85.49%)	1240492 (94.43%)
DoS	112012 (20.23%)	93843 (11.29%)	61676 (4.70%)
Fuzzing	24537 (4.43%)	20782 (2.50%)	7205 (0.55%)
RPM	1740 (0.31%)	3333 (0.40%)	2222 (0.17%)
Gear	1600 (0.29%)	2666 (0.32%)	2000 (0.15%)

Fuzzing Attack. An attacker injects a large number of CAN messages using arbitrary CAN IDs during fuzzing attacks. The *randint* function was used to generate random numbers regarding the fuzzing attack. We use random CAN IDs between 0x000 and 0x7FF, arbitrary lengths, and data for a 5-seconds period and a 2-seconds duration, and the interval time was 1 ms. The attack started at 20 seconds.

Spoofing Injection Attack. We injected *RPM* and *Gear* spoofing attacks into the spoofing attack dataset. We injected the spoofing attack for a 5-seconds period and a 2-seconds duration, the injection started from 20 seconds. We detail the spoofed data as follows, where xx and yy are spoofed values, and CS is a checksum value. Regarding spoofing attack for the Toyota, we used CAN ID 1C4, DLC 8, data xx, yy, 00, 00, 00, 00, 00, CS as the *RPM* data with 23 ms interval, and CAN ID 3BC, DLC = 8, data 00, xx, 00, 00, 00, 00, 00, 00 as the *Gear* data with 100 ms interval. Regarding the spoofing attack for the Subaru, we used CAN ID 141, DLC 8, data 00, 00, 00, 00, yy, xx, 02 as the *RPM* data with 10 ms interval, and CAN ID 148, DLC = 8, data xx, 00, 00, 01, 00, 00, yy, 01 as the *Gear* data with 10 ms interval. For the spoofing attack of the Suzuki, we used CAN ID 13F, DLC 8, data xx, yy, 00, 00, 00, 00, 00, 00 as the *RPM* data with 20 ms interval, and CAN ID 381, DLC = 5, data 00, xx, 00, 00, 00 as the *Gear* data with 100 ms interval.

Timestamp	CAN ID	DLC	D1	D2	D3	D4	D5	D6	D7	D8	Label
12.993251	141	8	53	26	E9	28	79	84	20	2	Benign
12.993385	376	1	FA	-1	-1	-1	-1	-1	-1	-1	Benign
12.998457	148	8	0B	13	AC	3	41	00	11	1	Benign
12.998699	149	8	79	1	3F	97	AA	34	00	00	Benign
12.999577	2	8	F1	DB	70	3	3F	00	00	00	Benign
12.999777	0	8	00	00	00	00	00	00	00	00	DoS
13.000077	0	8	00	00	00	00	00	00	00	00	DoS
13.000377	0	8	00	00	00	00	00	00	00	00	DoS
13.000677	0	8	00	00	00	00	00	00	00	00	DoS
13.000977	0	8	00	00	00	00	00	00	00	00	DoS
13.001277	0	8	00	00	00	00	00	00	00	00	DoS
13.001577	0	8	00	00	00	00	00	00	00	00	DoS
13.001915	280	8	3	6	10	00	00	00	00	00	Benign
13.002115	0	8	00	00	00	00	00	00	00	00	DoS
13.002415	0	8	00	00	00	00	00	00	00	00	DoS
13.002730	140	8	00	44	73	44	00	00	1F	20	Benign
13.002958	141	8	53	26	E9	28	79	84	20	2	Benign

(a) **DoS** Dataset

Timestamp	CAN ID	DLC	D1	D2	D3	D4	D5	D6	D7	D8	Label
12.999577	2	8	F1	DB	70	3	3F	00	00	00	Benign
13.000000	535	8	33	F1	46	DD	0D	56	BC	F0	Fuzzing
13.001000	1b6	8	EC	8D	D9	A2	37	41	5D	90	Fuzzing
13.001915	280	8	3	6	10	00	00	00	00	00	Benign
13.002730	140	8	00	44	73	44	00	00	1F	20	Benign
13.002958	141	8	53	26	E9	28	79	84	20	2	Benign
13.003196	144	8	00	00	5C	2	1	34	1	44	Benign
13.003440	152	8	E9	6C	00	00	00	00	1	80	Benign
13.004000	75	8	DE	A6	77	5F	10	C1	8	92	Fuzzing
13.005000	293	8	67	80	CF	37	B8	99	25	2B	Fuzzing
13.006000	749	8	54	71	47	EC	F7	CC	2C	F7	Fuzzing
13.006594	370	8	E4	00	00	40	CF	0F	00	00	Benign
13.007446	15A	8	80	1A	2	00	00	00	00	12	Benign

(b) **Fuzzing** Dataset

Figure 16. Examples of NAIST In-vehicle CAN Attack Dataset

5.4 Chapter Summary

In this chapter, we discuss the attack scenarios and methodologies of our developed NAIST CAN attack datasets. We make use of DoS, Fuzzing, and Spoofing (Speed, Handle angel, Gear, and RPM) attack scenarios for the dataset development. We use the aforementioned datasets in our experiments in Chapters 6 & 7.

6. LSTM-based In-vehicle CAN bus Intrusion Detection System

6.1 Introduction

The Controller Area Network (CAN) bus protocol is one the most important transformations introduced to the car industry. Developed by Robert Bosch in the 1980s, the CAN is an International Standardization Organization (ISO) defined serial communication bus that is in charge of the flow of information between the Electronic Control Units (ECUs) of a car. In simpler words, the CAN bus coordinates the movements between the engine, the brakes, the steering wheel, etc., i.e., it makes the modern car *connected*. The CAN protocol was initially engineered for industrial machinery, however it has been adopted for vehicular network communications.

The modern car is comprised of about 50 to 100 ECUs, some of which are connected through the CAN bus. The CAN bus protocol is effective for vehicular network systems because of its low cost and centralized system. The ECUs communicate with messages by using the CAN protocol. Each ECU receives messages with unique CAN bus IDs which are used for intra-interactions. Fig. 1 shows the 11 bit mode CAN message format.

Despite its importance, the CAN bus network is designed without security features, making it susceptible to confidentiality, integrity, and availability attacks. In in-vehicle communication systems, messages are transferred to the vehicle system managed by the CAN bus protocol. Hundreds of sensor data communicate to send messages to the CAN bus system. An ECU can share control data with an outside element of the vehicle through a network system. The later possibility increases the attack surface of the CAN bus protocol [25, 58]. The main security issues of the CAN bus arise because it broadcasts all the ECUs' messages without encryption nor authentication[48]. Consequently, ECUs are vulnerable to basic hacking techniques; thus, attackers can easily take control of the car system and cause great damage.

Koscher et al. [42] demonstrated that it is possible to compromise the CAN bus system and ECUs by investigating wireless attacks into the vehicle system.

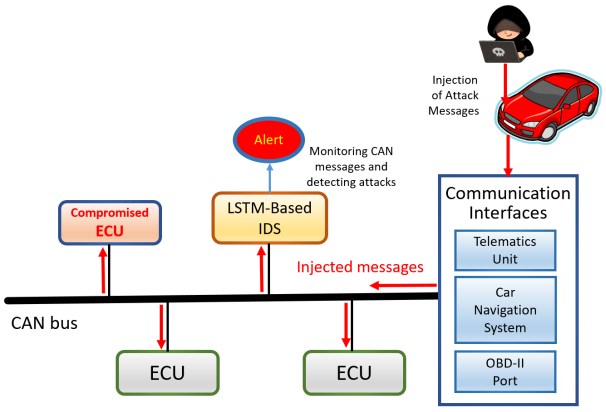


Figure 17. Typical Architecture of Intrusion Detection for CAN bus Network. The IDS monitors the messages exchanged in the CAN bus and gives an alert if it encounters suspicious activities.

They examined how CAN messages can be susceptible to Spoofing, and to what extent the CAN bus protocol is vulnerable to Denial of Service (DoS) attacks.

There are two obvious solutions for thwarting the attacks on the CAN bus system: introducing a backward-compatible authentication mechanism or developing an Intrusion Detection System (IDS). In this chapter, as an extension of our previous work [28] where we developed a CAN attack dataset consisting of DoS, Fuzzing and Spoofing, we opt for the second option. We propose Long Short-term Memory (LSTM)-based IDS for detecting attacks in the CAN bus system of a vehicle. In this extension, we obtain more fine-grained results and we compare our method to the Survival Analysis method [25]. We provide more details later in this section. LSTM is a powerful deep learning classifier that was created to address the look-back-in-time issue of Recurrent Neural Networks (RNN). We employ a deep learning algorithm because artificial intelligence (AI) is the contemporaneous dominant technology with proven applications in various fields such as image recognition, voice recognition, weather forecasting, market analysis, etc., [49].

An IDS can play an essential role in regards to cyber-attack detection and mitigation. A traditional IDS is unable to detect malfunction attacks on the CAN bus; thus, it is challenging for it to distinguish unknown attacks. Based on the requirements and functionalities, there are different types of IDS: signature-

based, anomaly-based, misuse-based, and hybrid [24, 44]. A signature-based IDS is unable to detect unknown attacks, whereas an anomaly-based IDS is capable of detecting unknown and malfunction attacks.

Fig. 17 shows the typical IDS architecture regarding in-vehicle network attacks. By using an external connection such as an OBD-II diagnostics port, a telematics unit, or in-vehicle infotainment (IVI), an attacker may inject an attack into the real car. An IDS can be placed in-between the CAN bus and the external connection. It will be responsible for filtering all the traffic into the CAN bus system; the IDS will send an alert message when an attacker attempts to inject malicious traffic.

In our setup, we collect raw CAN bus messages from a real car, and we inject attacks to develop our own DoS, Fuzzing, and Spoofing datasets. We train our LSTM model with benign and attack classes. The proposed model effectively classifies benign and attack instances with a high detection accuracy of 99.995% and with low false positive and false negative detection rates. We experiment with both binary and multiclass classification models. We also investigate the CAN bus attack detection performance by hyper-parameter values tuning. Based on the systematic experimentation, we select the best hyperparameter values to develop a robust IDS regarding the CAN bus attack detection. Our experiment results provide better directions on how fine-tuned hyper-parameter values significantly affect the detection accuracy and overall performance. We evaluate our model's performance based on F1 score, AUC-ROC curve, and false positive and false negative rates. We also evaluate our method against the Survival Analysis Dataset for automobile IDS [25] which was developed by the Hacking and Countermeasure Research Lab, Korea. The proposed model provides reasonable detection accuracy and detection rates against the Survival Analysis Datasets and the LSTM model achieves a higher detection rate compared to the Survival Analysis method.

In this chapter, our major contributions are as follows:

- We develop CAN system attacks (DoS, Fuzzing, Spoofing) datasets by using the CAN messages of a real car.
- To the best of our knowledge, we are the first to propose an effective LSTM-based IDS for in-vehicle CAN bus systems to detect well-known network

attacks: DoS, Fuzzing and Spoofing.

- We provide an effective pre-processing method to develop an effective LSTM-based supervised classification model regarding the CAN bus attack detection.
- We select the best hyper-parameter values to develop an effective CAN bus IDS based on LSTM.

6.2 Related Works

In this section, we discuss the essence of the related work regarding network anomaly detection of in-vehicle systems.

Koscher et al. are the first to demonstrate attack injection through wireless communication in in-vehicle network systems during an investigation of the security of modern vehicles [42]. They employed the CARSHARK tool to debunk numerous security vulnerabilities on the CAN bus, and they showed that the CAN broadcasting characteristics, when applied to all nodes, makes it easy for an attacker to intrude into the communication messages. Kleberger et al. investigated the security threats and attacks of in-vehicle network systems, they afterwards discussed the problems and solutions [41]. They also argued about IDS and architectural security features. Loukas et al. [46] proposed a deep learning-based Intrusion Detection System for the in-vehicle network systems by leveraging several machine learning classifiers. They conducted their experiment by injecting cloud-based attacks to a robotic vehicle. Their results show that LSTM is more suitable for in-vehicle intrusion detection with an overall accuracy of 86.9%. Kim et al. [58] proposed Generative Adversarial Networks (GAN)-based IDS (GIDS) for in-vehicle networks. They studied four vehicular attacks: DoS, Fuzzy[1], RPM, and Gear. As per their experiment results for the second discriminator in GIDS, they obtained attack detection rates of 99.60%, 99.50%, 99.00%, and 96.50%, respectively. Han et al. proposed a survival analysis method regarding intrusion detection for vehicular networks [25]. They extracted attack data from a real car, and they also injected attacks to generate: Flooding, Fuzzy, and Malfunction

[1]We use "Fuzzy attack" to keep the same vocabulary as in the related work, but we consider "Fuzzy attack" and "Fuzzing attack" to be the same in this chapter.

attack datasets. Kang et al. [34] devised an unsupervised deep belief network (DBN)-based IDS for the CAN BUS. They produced attack datasets by using a packet generator named Open Car Test-bed and Network Experiments (OCTANE). In paper [24], the authors studied several kinds of attacks against the connected cars and provided an overview of Artificial Neural Networks (ANN)-based IDS to mitigate cyber attacks on modern vehicle systems. The researchers in [32] used different machine learning algorithms to classify CAN bus messages. Their results show that the k-nearest neighbor (k-NN) algorithm performed better with an accuracy of 86.00%. Song et al. [62] proposed an IDS for in-vehicle networks based on a time interval analysis (TIA) of the CAN messages.

L. Hyunsang et al. [43] developed an IDS by analyzing the request-response message in the CAN bus, based on an offset ratio and time interval analysis. Khan et al. [35] Khan et al. investigated SDN-based false data injection into the brake-related ECUs. They developed false information attack dataset and applied LSTM to detect the attack, and they achieved a detection rate of 87%. Woo et al. [75] developed an in-vehicle CAN security protocol and analyzed the wireless attacks on the connected car. They discussed the connected car environment, several kinds of attack models, and security requirements. Taylor et al. [66] engineered an IDS based on the LSTM model. Their proposal relies on the prediction of the next data of the CAN bus network, while acknowledging that the data originates from the senders.

6.3 Attacks Used in the Model

We mainly experiment with three types of attacks in this chapter: DoS, Fuzzing, and Spoofing. Table 14 shows the NAIST CAN attack dataset details which we consider for further experimentation. The attack mentioned above, attack scenarios, and the dataset creation details are available in Section 5.1 and 5.2. We also study with the Survival Analysis dataset [25], dataset details available in section 7.3.1.

Table 16. Survival Analysis Dataset for automobile IDS - Benign and Attack Instances

Type of Attacks	Sonata	Soul	Spark
Benign	468527	717489	366510
Flooding	32422	33141	22587
Fuzzy	18118	39812	5812
Malfunction	15974	7401	8047

6.3.1 Survival Analysis Dataset for automobile IDS - Benign and Attack Instances

Regarding the Survival Analysis datasets multiclass classification, we concatenate all the three attack classes for each car model. After concatenation, we observe from Table 16 that Sonata contains 468527 benign elements and Flooding, Fuzzy and Malfunction (Malf.) elements are 32422, 18118 and 15974, respectively. The Soul car's benign elements are 717489, whereas its Flooding, Fuzzy and Malfunction elements are 33141, 39812 and 7401, respectively. The Spark car model has 366510 benign elements and 22587, 5812 and 8047 of Flooding, Fuzzy and Malfunction elements, respectively. We observe that the number of benign class elements is higher compared to the number of attack class elements. As per Table 16, 89.44% of the elements belong to the benign class, whereas we have 5.08%, 3.67% and 1.81% of Flooding, Fuzzy and Malfunction elements, respectively.

6.4 LSTM-based Network Intrusion Detection System

For our investigation, we use python PyCharm IDE 2019.2.2 and Keras [2] with TensorFlow as backend. We conduct our experiment with Intel Core i7 CPU 2.20 GHz, 16 GB RAM, Windows 10 (64-bit), and NVIDIA GeForce GTX 1050. We use the *categorical_cross entropy* as loss function. The *Nadam* optimizer is applied with a learning rate of 0.0001, the rest of the parameters conserve their default values and *softmax* is used as an activation function output. Tables 17 and 18 provide the experimental settings regarding attacks detection based on LSTM binary and multiclass classification models. We preprocessed raw CAN dataset before inputting the model, we train the model by providing 80% of the

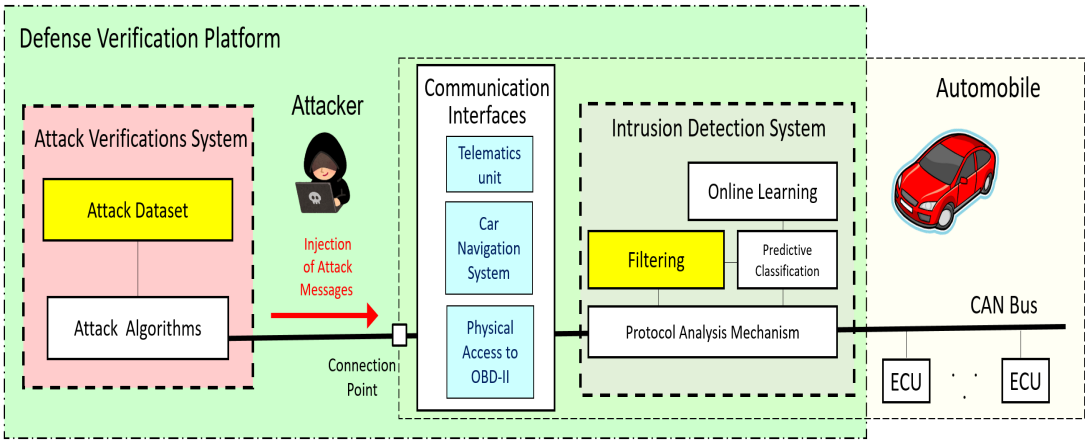


Figure 18. CAN bus Network System Defense Verification Platform. Consist of Two modules: Attack Verification and Intrusion Detection System

elements and we test the classifier with 20% of the elements. After training, the classifier can classify the attack and benign class elements. Fig. 21 schematizes the deep neural network model wherein we can input the CAN bus data into the input layer and, after preprocessing, the classifier will provide the output as a benign or attack class.

Figure 18 depicts the architecture of the proposed defense verification platform about in-vehicle CAN bus and ECUs. Two modules compose the architecture: the attack verification platform and the Intrusion Detection System. As per the attack verification platform, we extract raw CAN bus data from the real car. We develop the attack datasets by making use of our attack creation algorithm. As per the connection point, an attacker can compromise the CAN bus network system by using the communication interfaces: Telematics unit, Car Navigation System, Physical access to an OBD-II port. We place an IDS into the CAN bus system, the IDS filter module filters all the malicious traffic into the CAN bus message communication and provides an alert message in case of malicious traffic injection.

6.4.1 Dataset Preprocessing

For our experiment, we use NAIST CAN attack labelled dataset to evaluate the performance of our proposed LSTM model. We extract the attack-free dataset from a Toyota Hybrid car and we develop attack datasets –DoS, Fuzzing and Spoofing– by injecting attacks through a program written with the Python programming language.

In Section 5.2, we discuss in detail about the NAIST CAN attack datasets. We use the Vehicle Spy3 Professional tool to extract the raw data from a real car. The NAIST CAN attack raw dataset only consists of the attack-free instances extracted from the Toyota Hybrid car. For the attack dataset, we develop a Python-based program for injecting attacks to generate DoS, Fuzzing, and Spoofing datasets. The CAN bus raw dataset is in hexadecimal format shown in Fig. 13; we experiment with the attack dataset without decoding and converting hexadecimal to decimal as per machine learning requirement. In the CAN message format, the classification label R represents the benign class message and T denotes the injected attack message. In our experiment, we switch the roles as follows: R as benign and T as an attack class name. In our experiment, we only took into account 10 features in the dataset –CAN ID, DLC, Data [D0-D7]– and the Label column. CAN ID is an identifier of the CAN messages. DLC data bytes is between 0-8. The data field contains 64 bits, and we position each byte in specific columns such as D0-D7. Since we did not consider time interval analysis to detect the intrusion, we did not analyze the timestamp field.

We concatenate three attack datasets and apply the multiclass classification. From Table 14, the number of benign instances is 947931 (69.22%), DoS instances, 286502 (20.92%), Fuzzing instances, 114027 (8.33%), and Spoofing instances, 21072 (1.54%). We observe that the number of benign instances is higher than the number of attack instances. CAN ID data fields range from 1-8 bytes. As per machine learning requirement, we fill up the CAN bus data fields with blank values by -1, which helps to keep the integrity of the datasets intact. We use 10 features that we label as benign, DoS, Fuzzing, and Spoofing attacks. We use 80% of the data for training, and the remaining 20% of the data is used for the testing set. There are 1095625 instances in the training set and 273907 instances in the testing set.

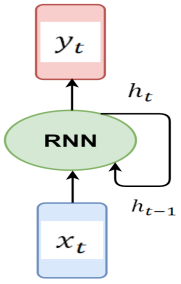


Figure 19. Basic RNN Architecture

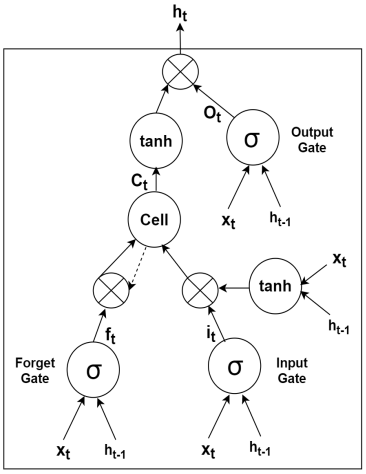


Figure 20. LSTM Cell Architecture

6.4.2 Application of the Long Short-Term Memory (LSTM) Model

LSTM is a special kind of recurrent neural networks. It was introduced by Hochreiter and Schmidhuber in 1997 [27]. We contend that LSTM is suitable for this research because the LSTM performs well regarding Time Series data and Sequence Classification.

As per the recurrent neural network (RNN) architecture (Fig. 19), we can process a sequence of data $x_1, ...x_n$ by applying RNN and it will produce a sequence of outputs $y_1, ...y_i$.

$$h_t = f_W(h_{t-1}, x_t) \tag{6}$$

$$h_t = \text{New state}$$

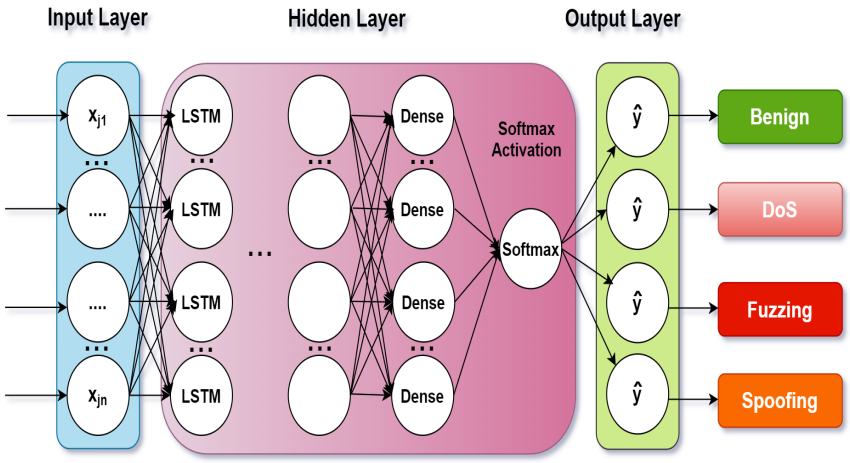


Figure 21. LSTM IDS Architecture Regarding the Attack Classification

$$f_W = \text{Function with parameter W}$$
$$h_{t-1} = \text{old state}$$
$$x_t = \text{Input vector at time step t}$$

By applying the recurrence equation, Equation 1, at every time step, we can process a sequence of vectors $x_1, ...x_n$. The same function and set of parameters $f_W(h_{t-1}, x_t)$ are used in every time step t with the input x_t and the old state h_{t-1} and, we get as output h_t, new state.

The standard recurrent sigma cell's mathematical expressions are as follows:

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b) \tag{7}$$

$$y_t = h_t \tag{8}$$

Where x_t represents the input, h_t the recurrent information, and y_t the output of the cell at time t, W_h and W_x are the weights and b is the bias.

However, standard recurrent cells of the recurrent networks are incompetent to handle long-term dependencies; since the gap between the associated inputs grows, it is complicated to learn the connection information. Hochreiter and Schmidhuber (1997) proposed the LSTM cell to overcome the "long-term dependencies" problem. They introduced the "gate" into the cell; thus, it facilitates

the standard recurrent cell to retain memory. Generally, the LSTM cell signifies LSTM with a forget gate [76].

Three gates are available in LSTM and they are responsible for the cell state protection and control [1]. Figure 20 depicts the LSTM cell architecture consisting of the input vector x_t, hidden input vector h_{t-1} and output vector h_t.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{9}$$

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{10}$$

$$\bar{C_t} = tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{11}$$

$$C_t = f_t * C_{t-1} + i_t * \bar{C_t} \tag{12}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{13}$$

$$h_t = o_t * tanh(C_t) \tag{14}$$

W_f, W_i and W_c are weights, and b_f, b_i and b_c are bias. The sigmoid layer makes the decision, it is called the "forget gate layer" and it outputs between 0 and 1.

C_t is the new cell state, it is obtained from the old cell state C_{t-1} and is regulated by the input i_t and forget f_t gates.

We employ the LSTM model for supervised binary and multiclass classification. Regarding supervised learning, we have input variables (X) and an output variables (Y). We use the LSTM model to learn the mapping function from the input to the output: $Y = f(X)$. The objective is to determine the mapping accurately; hence, we can predict the output variables (Y) when we input the new input data (X).

$$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ . \\ . \\ . \\ X_i \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & ... & x_{1n} \\ x_{21} & x_{22} & x_{23} & ... & x_{2n} \\ x_{31} & x_{32} & x_{33} & ... & x_{3n} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ x_{i1} & x_{i2} & x_{i3} & ... & x_{in} \end{bmatrix}$$

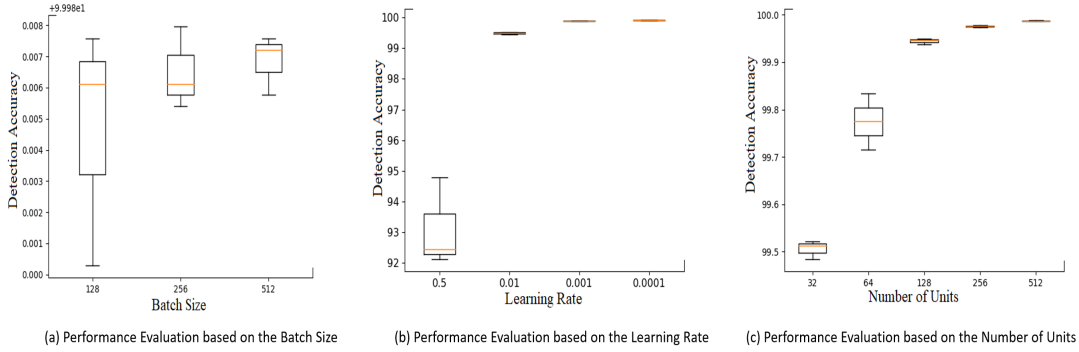
$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_i \end{bmatrix}$$

We give an input sequence $X = (X_1, X_2, X_3, \ldots, X_i)$ and $X_j = (x_{j1}, x_{j2}, x_{j3}, \ldots, x_{jn})$, $j \in [1, \ldots, i]$, where x_{jz}, $z \in [1, \ldots, n]$, is an element of the input variables (X), and we obtain an output sequence $Y = (y_1, y_2, y_3, \ldots, y_i)$, where y_q, $q \in [1, \ldots, i]$, is an element of the output variables (Y). For each time step, the input X_j is each row/CAN packet payloads of the corresponding CAN ID of the dataset. Each row/CAN packet is an observation comprised of ten features as input variables (X) and one output variable to be predicted (Y).

We have to reshape the input because the input shape for the LSTM model must be three-dimensional (Samples, Time Steps, and Features). We use single time steps and define a 3D array of the LSTM input layer regarding the fitting of the model and when making the predictions.

We experiment with changing the different hyper-parameter values such as optimizer, learning rate, units, activation function, etc. We select the best hyper-parameter values based on the systematic experimentation for our experiment to achieve the best detection accuracy.

We train our LSTM classifier with benign and CAN bus attack instances; 80% of the dataset is used for training the classifier, and 20% is used for the testing. We evaluate to which degree the classifier is capable of detecting the attack instances. We conduct our experiment with a Vanilla LSTM model, and we also use the Stacked LSTM model with single to five hidden layers and arbitrary unit settings: 512-512-256-128-64, and we apply binary and multiclass classifications. As per Fig. 27(c), we observe that it is better to use bigger units –512-256– instead of lower units to achieve the best detection accuracy with low variance. Fig. 27(a) shows that 256-512 batch size provides better detection accuracy. Table 17 and Table 18 describe the LSTM parameter details which we use for our experiment. We use a single to five LSTM stacked layers on top of each other; the final result is a dense layer with *softmax* activation, which gets input from the last layer output, the LSTM layer.



(a) Performance Evaluation based on the Batch Size (b) Performance Evaluation based on the Learning Rate (c) Performance Evaluation based on the Number of Units

Figure 22. Performance Evaluation based on the Batch Size, Learning Rate and Number of LSTM Units

In this experiment, we use the Keras [2] API. Keras is written in Python; hence, it is easy to use it along with Tensorflow. Table 17 provides the parameter values that we use in our binary classification experiment. We select the best hyper-parameters values by experimenting with hyper-parameter value changes. We utilize the *Adam* optimizer with a learning rate of 0.0001. We employ the *binary_crossentropy* as the loss function and the *sigmoid* as an output activation function. In the proposed multiclass LSTM model experiment, we use the *Nadam* optimizer. We use the optimizer's learning rate of 0.0001, and the remaining

Table 17. LSTM Parameters for Binary Classification

Parameters	**Value**
Activation Function Input	tanh
Epoch	200
Activation Function Output	sigmoid
Optimizer	Adam
Learning Rate	0.0001
Batch Size	512
Loss Function	binary_crossentropy
Encoder	Label Encoder

Table 18. LSTM Parameters for Multiclass Classification

Parameters	**Value**
Activation Function Input	sigmoid
Epoch	200
Activation Function Output	softmax
Optimizer	Nadam
Learning Rate	0.0001
Batch Size	512
Loss Function	categorical_crossentropy
Encoder	Label Encoder

parameters of the optimizer are used with their default values. In Fig. 27(b), we see that 0.0001 provides the highest detection accuracy with the lowest variance. The *categorical_crossentropy* optimizer is used as loss function. We set the number of iterations to 200 epochs.

We perform a validation through the *fit()* function by using validation data. After training and testing, we calculate the accuracy and loss based on a number of correctly classified instances. It is challenging to optimize the model loss and achieve the best detection accuracy for the best hyper-parameter values selection. Hyper-parameter values are strong indicators for gaining better accuracy and detection rates when using a deep learning model. We have to find out the right optimizer, activation function, learning rate, and loss functions to develop a useful

model and efficient design. *Nadam* and *Adam* are both appropriate optimizers concerning large datasets and efficient models. We fine-tune the hyper-parameter values such as layers, activation functions, learning rates, and loss functions. We experiment with hype-parameter tuning by using layer 1 with 512 arbitrary units. In Section 7.5, we discuss the details regarding how hyper-parameter values change the experiment settings and results.

Table 19. Binary Classification Results - NAIST CAN Attack Dataset

Attack	**Acc**	**TPR**	**TNR**	**FPR**	**FNR**	**AUC**
DoS	100%	1.00	1.00	0.00	0.00	1.00
Fuzzing	99.98%	0.9993	1.00	0.00002	0.0007	1.00
Spoofing	100%	1.00	1.00	0.00	0.00	1.00

6.5 Experiment Results and Performance Evaluation

Performance measurement is an essential aspect in machine learning. We evaluate the CAN bus network attack detection performance by using the detection accuracy, detection rate, Area Under The Curve (AUC)-Receiver Operating Characteristics (ROC) curve, and F1 scores [3].

We use the AUC-ROC curve to perform visualizations for multiclass classification at all the classification thresholds. We plot the True Positive Rate (TPR) against the False Positive Rate (FPR) in the AUC-ROC curve wherein FPR and TPR are on the x-axis and y-axis, respectively. A higher AUC-ROC is better because it demonstrates the strength of the model. In other words, the model is strong when AUC-ROC is close to 1.0, and the model is weak (worse model) when AUC-ROC is close to 0.0. Based on the number of classes, we can plot AUC-ROC curves for multiclass classification.

The F1 score is an essential factor for measuring machine learning performance evaluation when the datasets are imbalanced. In the case of an imbalanced dataset, we cannot evaluate the performance by only detecting the accuracy (Acc). Similar to AUC-ROC, the model is strong if the F1 score is around 1.0, and the model is weak if the F1 score is close to 0.0. The F1 score is the weighted average results of the precision and recall [61, 33].

We visualize the model's performance by using the AUC-ROC multiclass curve and also the class-specific multiclass curve. In the following section, we discuss the experimental results and the effectiveness and performance of the model based on hyper-parameter tuning.

We conduct our experiment with the binary classification parameter settings depicted in Table 17. Table 19 shows the binary classification results, we observe that DoS and Spoofing attacks are classified with 100% accuracy, and Fuzzing is detected with a 99.98% accuracy. FPR is zero for the DoS and Spoofing attacks, wherein few false positive and false negatives are available for the Fuzzing attack.

6.5.1 LSTM Layer(s) - Attacks Classification Experiment Results

There are two main hyper-parameters in Artificial Neural Networks which control the network's topology: the number of layers and number of nodes in the respective hidden layer. The most effective approach is to select the number of layers and nodes for each hidden layer and other hyper-parameter values, we have to proceed with systematic experimentation for the particular predictive models [15].

To achieve an effective detection accuracy and detection rate, we have to select and find out the LSTM layer's number providing the best accuracy and detection rates. We study a single to five LSTM layers based on the parameter settings in Table 18, and we compare the performance among them. In Table 20, according to our experiment results, Vanilla LSTM with a single hidden layer provides the best detection accuracy wherein the FPR and FNR are lower. Table 20 shows the layer-wise results wherein L1 provides the best detection accuracy, 99.995%, and when we increase the layer size to L2-L5, we notice a decrease of the detection accuracy and an increase of the false positive and false negative rates.

Fig. 24 shows the LSTM multiclass confusion matrix (CM). For the Nadam optimizer, which we consider for this research, we observe that DoS and Spoofing attacks are accurately classified (100%), but few false negatives (14 instances) are available regarding the Fuzzing attack. Based on the CM and Table 21 (L1), we detect DoS and Spoofing attacks more accurately without false positives and false negatives wherein the FPR and the FNR are 0.00004 and 0.0002, respectively. Still, some false negatives are available regarding Fuzzing attack detection. Our

Table 20. LSTM Layer(s) Multiclass Classification Results - NAIST CAN Attack Dataset

Layer	Attack	Acc	Recall	F1	FPR	FNR
L1	Benign	**99.995%**	1.0000	1.0000	0.0002	0.0000
	DoS		1.0000	1.0000	0.0000	0.0000
	Fuzzing		0.9994	0.9997	0.0000	0.0006
	Spoofing		1.0000	1.0000	0.0000	0.0000
	Avg		0.9998	0.9999	0.00004	0.0002
L2	Benign	99.988%	1.0000	0.9999	0.0004	0.0000
	DoS		1.0000	1.0000	0.0000	0.0000
	Fuzzing		0.9986	0.9993	0.0000	0.0014
	Spoofing		1.0000	1.0000	0.0000	0.0000
	Avg		0.9997	0.9998	0.0001	0.0003
L3	Benign	99.978%	1.0000	0.9998	0.0007	0.0000
	DoS		1.0000	1.0000	0.0000	0.0000
	Fuzzing		0.9974	0.9987	0.0000	0.0026
	Spoof		1.0000	1.0000	0.0000	0.0000
	Avg		0.9993	0.9996	0.0002	0.0007
L4	Benign	99.987%	1.0000	0.9999	0.0004	0.0000
	DoS		1.0000	1.0000	0.0000	0.0000
	Fuzzing		0.9985	0.9992	0.0000	0.0015
	Spoofing		1.0000	1.0000	0.0000	0.0000
	Avg		0.9996	0.9998	0.0001	0.0004
L5	Benign	99.989%	1.0000	0.9999	0.0003	0.0000
	DoS		1.0000	1.0000	0.0000	0.0000
	Fuzzing		0.9990	0.9993	0.00004	0.0010
	Spoofing		1.0000	1.0000	0.0000	0.0000
	Avg		0.9997	0.9998	0.0001	0.0003

model's overall detection accuracy and the detection rate are adequate to classify the CAN bus network attacks efficiently. We observe that L1 provides the best detection rate regarding the Fuzzing attack detection wherein L2-L5 decrease the

Table 21. Layer(s) Multiclass Classification results - NAIST CAN Attack Dataset

Layer	Accuracy	TPR	TNR	FPR	FNR
L1	**99.995%**	0.9998	1.0000	0.00004	0.0002
L2	99.988%	0.9997	0.9999	0.0001	0.0003
L3	99.978%	0.9993	0.9998	0.0002	0.0007
L4	99.987%	0.9996	0.9999	0.0001	0.0004
L5	99.989%	0.9997	0.9999	0.0001	0.0003

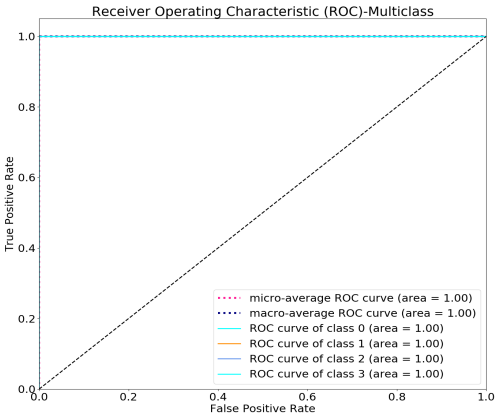


Figure 23. NAIST CAN Attack Dataset Classification Receiver Operating Characteristics (ROC)

detection rate. That is why it is better to use L1 for CAN bus attack detection; additionally, it requires less computation cost compared to L2-L5 layers. Fig. 23 depicts the ROC curve, and we observe that AUC is 1.0 for all the classes, which indicates that all the instances are almost classified accurately, it demonstrates that our proposed model is quite effective to classify the CAN bus network attacks.

We assume that in case of using the 30% testing data set. The FPR and FNR may increase slightly based on a large testing set, and also it may degrade the detection accuracy a bit.

6.5.2 Nadam Learning Rate - LSTM Classification Results

Gradient descent is one of the most popular and extensively used optimizer algorithms for optimizing neural networks [55]. The optimizer learning rate (LR) is one of the critical factors regarding detection accuracy. Weights and bias are radical changes in the case of the usage of a large learning rate. A large learning rate may cause an overpass of the global minima. To avoid the risk of an overpass, it is better to set the minima to a smaller learning rate instead of a large value. The training period is longer and it also proliferates the time to converge to a small learning rate [17, 22]. We compare the performance by using a few learning rate values. Table 22 shows that the lower the learning rate, the better the detection accuracy. We observe that a learning rate of 0.0001 achieves a detection accuracy of 99.995% whereas a 0.5 learning rate's detection accuracy is 91.57%, and an LR of 0.5 cannot classify any attack instances. An LR of 0.0001 provides the best detection accuracy because a smaller learning rate can thoroughly learn the dataset and be able to classify the instances accurately.

Table 22. Nadam Learning Rate LSTM Classification Results - NAIST CAN Attack Dataset

Learning Rate	**Acc**	**Recall**	**F1**	**FPR**	**FNR**
0.0001	**99.995%**	0.9998	0.9999	0.00004	0.0002
0.001	99.87%	0.9981	0.9979	0.0006	0.0019
0.01	99.18%	0.9868	0.9862	0.0043	0.0132
0.5	91.57%	0.9317	0.7844	0.0273	0.0683

6.5.3 Optimizers - Classification Results

Regarding the deep learning model performance optimization, we optimize the neural network by using one of the most popular algorithms, which is gradient descent. There are three different gradient descent algorithms: batch gradient descent, stochastic gradient descent, and mini-batch gradient descent. We have to select the gradient descent as per the amount of data we would like to compute the gradient of the objective function [55].

To achieve better accuracy, we need to select the right optimizer(s) for the model. We study six optimizers: *RMSprop*, *Adam*, *Adagrad*, *Adadelta*, *Adamax* and *Nadam*. As per Table 23, *Adam* and *Nadam* provide high detection accuracy regarding CAN bus attacks. We found that *Adam* and *Nadam* are appropriate optimizers regarding binary and multiclass classifications for CAN bus network attack detection. Both optimizers provide the best classification accuracy [19, 78, 40]. *Nadam* and *Adam* provides the detection accuracy of 99.995% and 99.993% and the detection rate for both optimizers is 0.9998. We observe that *Adagrad* and *Adadelta* optimzers provide lower detection accuracy as compared to other optimizers. As per Figure 24, we observe that *Nadam* and *Adam* classify most of the instances correctly with fewer false positives and false negative instances.

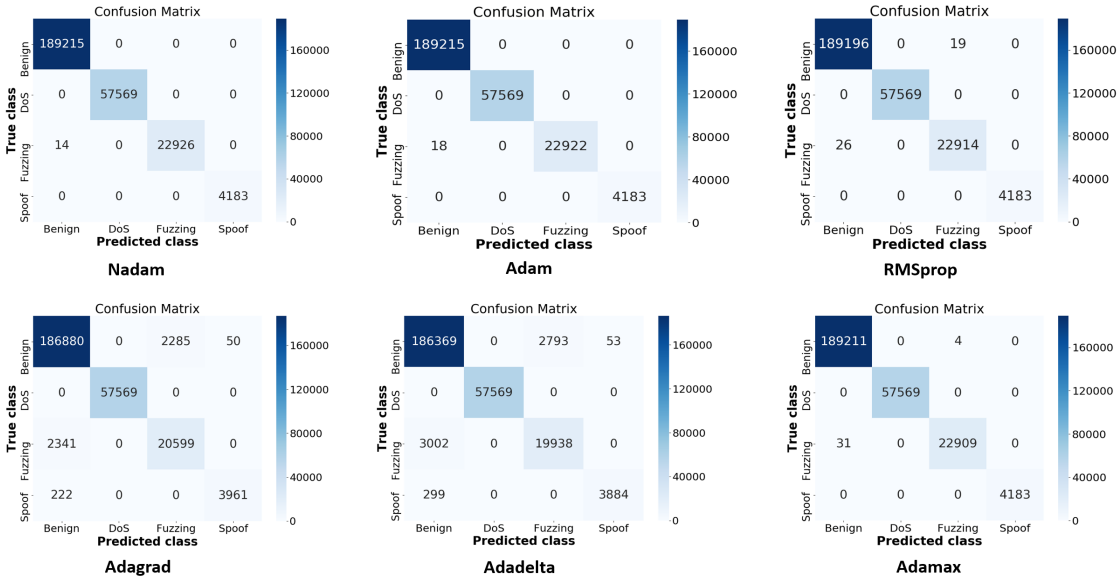


Figure 24. Gradient Descent Optimizer Confusion Matrix - NAIST CAN Attack Dataset

6.5.4 Activation Function - Classification Results

We use *sigmoid* as an input activation function and *softmax* as an output activation function for classifying the CAN bus attacks in our model. We study with three input activation functions by using L1 and we use the parameter values

Table 23. Optimizers Classification Results - NAIST CAN Attack Dataset

Optimizer	Accuracy	Recall	F1	FPR	FNR
RMSprop	99.984%	0.9997	0.9997	0.0001	0.0003
Adam	**99.993%**	0.9998	0.9999	0.00005	0.0002
Nadam	**99.995%**	0.9998	0.9999	0.00004	0.0002
Adagrad	98.212%	0.9581	0.9632	0.0099	0.0419
Adadelta	97.756%	0.9457	0.9534	0.0126	0.0543
Adamax	99.987%	0.9997	0.9998	0.0001	0.0003

Table 24. Activation Function-wise Classification Results - NAIST CAN Attack Dataset

Activation Function	Accuracy	Recall	F1	FPR	FNR
sigmoid	**99.995%**	0.9998	0.9999	0.00004	0.0002
relu	99.988%	0.9998	0.9998	0.00007	0.0002
tanh	**99.994%**	0.9998	0.9999	0.00005	0.0002

settings in Table 18. In this experiment, *softmax* output activation function remains the same, we only change the input activation functions as *tanh*, *relu* and *sigmoid*. Table 24 shows that *sigmoid* and *tanh* provide the best performances compared to *relu*.

The *sigmoid* and *tanh* input activation functions provide the best detection accuracy of 99.995% and 99.994% whereas *relu* provides a detection accuracy of 99.988%.

6.5.5 Loss Function - Classification Results

In a deep learning approach, the loss function is an essential factor that significantly impacts the detection accuracy. To optimize the model and reduce the error, we need to select the proper loss function for the model to predict the results accurately. We have to consider various factors before choosing the loss function. Based on the specific predictive model such as regression or classification losses, we have to choose the appropriate loss function [11].

We also change the loss functions and compare the performance among them. Table 25 shows that *categorical_crossentropy* and *kullback_leibler_divergence* perform well compared to the other optimizers regarding CAN bus attacks detection. Both optimizers provide similar detection accuracy of 99.995%.

Table 25. Loss Function-wise Classification Results - NAIST CAN Attack Dataset

Loss Function	**Acc**	**Recall**	**F1**	**FPR**	**FNR**
categorical _crossentropy	**99.995%**	0.9998	0.9999	0.00004	0.0002
MAE	91.100%	0.6641	0.6830	0.0720	0.3359
MSE	99.990%	0.9997	0.9998	0.0001	0.0003
KL_divergence	**99.995%**	0.9998	0.9999	0.00004	0.0002

6.5.6 Results comparison with the Survival Analysis method/dataset

We apply the proposed LSTM IDS to the Survival Analysis Dataset for automobile IDS [25]. The datasets contain three different car models with Flooding, Fuzzy, and Malfunction attacks. We use binary and multiclass classification according to Tables 17 and 18 parameter settings. From the experiment results, we observe that in Tables 26 and 27 LSTM can detect the attacks with almost 100% accuracy wherein Flooding and Malfunction detection rate is almost 1.00 and few false positives are visible regarding the Fuzzy attacks. The proposed method is quite effective in classifying the attacks with a high detection rate and it achieves a higher detection rate than the Survival Analysis method's average detection rate (Fig. 25). Our classifier can classify the Flooding and Malfunction attacks with a high detection rate of 1.00 (Table 27). The proposed method's Fuzzy attack detection rate is also high compared to the conventional method but it is not as good as for Flooding and Malfunction attacks.

6.6 Discussion

Researchers whose work revolves around deep learning face issues related to the availability of real-time public datasets. We produce our CAN bus attacks

Table 26. LSTM Multicass Classification Results - Survival Analysis Dataset for automobile IDS

Model	Attack	Acc	Recall	F1	FPR	FNR
Sonata	Benign	**99.997%**	1.0000	1.0000	0.0001	0.00001
	Flooding		1.0000	1.0000	0.0000	0.0000
	Fuzzy		0.9995	0.9996	0.00001	0.0005
	Malf.		1.0000	1.0000	0.0000	0.0000
	Avg		0.9999	0.9999	0.00004	0.0001
Soul	Benign	99.707%	1.0000	0.9994	0.0116	0.0000
	Flooding		1.0000	1.0000	0.0000	0.0000
	Fuzzy		0.9469	0.9701	0.0003	0.0531
	Malf.		0.9697	0.9087	0.0015	0.0303
	Avg		0.9792	0.9696	0.0034	0.0208
Spark	Benign	99.953%	0.9998	0.9997	0.0027	0.0002
	Flooding		1.0000	1.0000	0.0000	0.0000
	Fuzzy		0.9823	0.9831	0.0002	0.0177
	Malf.		1.0000	1.0000	0.0000	0.0000
	Avg		0.9955	0.9957	0.0007	0.0045

datasets from an actual car. We extract raw data from the real car, and we generate three separate attack scenario datasets: DoS, Fuzzing, and Spoofing. These types of attacks are considered the most prominent in the CAN bus system in terms of damage capabilities. The connected car market is rapidly growing; hence, the interest of hackers is proportionally growing. CAN bus data security is of utmost importance regarding safe driving.

The CAN protocol broadcasts the messages to all nodes without encryption or authentication. Hence, attackers can easily inject malicious messages to the CAN bus system and disrupt the confidentiality, integrity, and availability of the said system. If an attacker succeeds in injecting an attack and taking control of the CAN bus system, they can stop the engine, disable the brakes, turn the lights on/off, etc. An efficient IDS can protect the CAN bus systems. We conduct research on hyper-parameter values tuning, and we observe how the de-

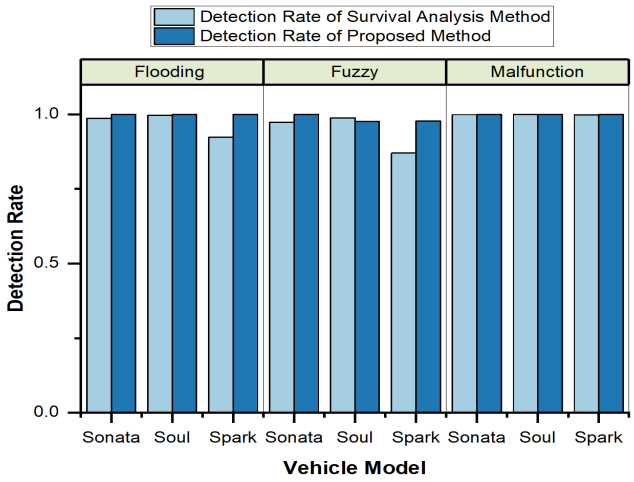


Figure 25. Detection Rate - Comparison with the Survival Analysis Dataset for automobile IDS

tection accuracy and detection rate vary depending on the hyper-parameter value changes. Our investigation results demonstrate that hyper-parameter values significantly affect the IDS detection accuracy. In this study, we were able to find the best hyper-parameter values for designing an effective Intrusion Detection System for the CAN bus network system. In this Investigation, we have taken multi-classification approach for detecting the type of attacks in CAN bus network. There might be a discussion that multi-label approach may be able to pick up several types of attacks if the attacks are mixed. But our focus was not in that direction in this research.

An effective IDS is essential regarding the in-vehicle CAN bus attack detection, and it will make a significant impact for the safe driving of the car. An anomaly-based IDS can play a vital role in mitigating the known and unknown attacks in the CAN bus network system. Preprocessing is one of the key facts regarding the IDS system to achieve the best performance. Our proposed preprocessing system is effective and it's working fine in LSTM for CAN bus network attack detection. We achieve the best detection rate regarding Toyota and Survival Analysis dataset experiment.

Our proposed LSTM-based IDS is effective and efficient. Without decoding the raw messages of the CAN bus, we can detect CAN bus network attacks. The

proposed model detects DoS and Spoofing attacks with a detection rate of 1.00, and the Fuzzing detection rate is 0.9994.

Table 27. LSTM Binary Classification Results - Survival Analysis Dataset for automobile IDS

Model	Attack	Acc	Recall	F1	FPR	FNR
Sonata	Flooding	100%	1.0000	1.0000	0.0000	0.0000
	Fuzzy	99.996%	1.0000	0.9999	0.00004	0.0000
	Malf.	100%	1.0000	1.0000	0.0000	0.0000
	Avg	99.999%	1.0000	1.0000	0.00001	0.0000
Soul	Flooding	100%	1.0000	1.0000	0.0000	0.0000
	Fuzzy	99.62%	0.9763	0.9880	0.0000	0.0237
	Malf.	100%	1.0000	1.0000	0.0000	0.0000
	Avg	99.87%	0.9921	0.9960	0.0000	0.0079
Spark	Flooding	100%	1.0000	1.0000	0.0000	0.0000
	Fuzzy	99.60%	0.9780	0.9780	0.0022	0.0220
	Malf.	100%	1.0000	1.0000	0.0000	0.0000
	Avg	99.87%	0.9927	0.9927	0.0007	0.0073

Although we use datasets from a specific vehicle, we believe that our model can work for the CAN bus system for any vehicle. To evaluate the performance of the proposed IDS, we also experiment with the Survival Analysis CAN bus attack dataset which is developed by the Hacking and Countermeasure Research Lab, Korea. The proposed LSTM models can classify the attack instances with high detection rates. We also compare our detection rate with the one of the Survival Analysis method, and our results show that LSTM is quite effective compared to the Survival Analysis method. As per our binary experiment results with the Survival Analysis datasets, we are able to detect Flooding and Malfunction attacks with 100% accuracy. We also observe the presence of few false negatives and false positives for the Fuzzy attacks. Our proposed LSTM model achieves higher detection accuracy compared with the conventional methods.

Hyper-parameter values selection is essential to develop a robust IDS by deep learning models. We select the hyper-parameter values based on the systematic

experimentation and fine-tuning of the values. The proposed hyper-parameter values are quite effective regarding in-vehicle CAN bus attack detection for the LSTM deep learning approach. We provide detailed experiment results regarding the hyper-parameter values tuning, and we select the best values from them. As per our experiment results, Layer 1 combined with the *Nadam* optimizer with a learning rate of 0.0001, the *categorcal_crossentropy* loss function, and the *Sigmoid* activation function provide the best detection accuracy regarding CAN bus IDS.

Our assumption is that the proposed LSTM model achieves a higher detection accuracy regarding DoS and Spoofing attacks because these attacks consist of specific CAN IDs with similar repeated attack patterns. The LSTM model is a robust deep learning algorithm, and after training, the proposed LSTM model can classify those attack traffic effortlessly. The Fuzzing attack dataset is developed based on the use of random CAN IDs and messages, and the attack pattern is almost similar to the legitimate traffic. Hence, the model faces difficulties in learning the intricate attack patterns, so the proposed model's Fuzzing attack detection rate is lower compared to the DoS and Spoofing attacks, and the classifier is unable to achieve 100% detection accuracy for the Fuzzing attack. The proposed LSTM model's performance regarding the detection accuracy may slightly degrade, and the FNR and FPR may increase when using a large testing set.

The major limitation of our model resides in the fact that we experiment in offline mode with labeled datasets, we are concerned regarding the performance of the IDS on online mode and also about the IDS effectiveness regarding unknown attacks detection. Additionally, we have not defined how to recover the vehicle system after injecting the attacks, i.e., the CAN bus system must be available even after our attack injections. Another key challenge is about how to embed the deep learning-based IDS with the CAN bus system.

In our future work, we will consider implementing this LSTM-based IDS to real CAN bus systems to evaluate IDS's performance in real-time. Furthermore, we will investigate how to detect unknown attacks in real vehicles. Finally, we will consider how to return the vehicle to a working condition after an injection attack incident.

6.7 Chapter Summary

In this chapter, we propose an effective long short-term memory (LSTM)-based Intrusion Detection System (IDS) for in-vehicle CAN bus network attack. We develop the CAN bus attack dataset by extracting the attack-free traffic from a real car. We generate attack datasets by injecting three kinds of attacks –DoS, Fuzzing, and Spoofing– into the attack-free dataset. We effectively preprocess the dataset, and our proposed LSTM model can classify benign and attack classes with a high accuracy of 99.995% and low false positive and false negative rates for the layer and optimizer we considered. We thoroughly study the hyper-parameter values changing, and we select the best parameter values to achieve efficient detection accuracy and detection rates. As per the experiment results, Vanilla LSTM provides the best detection accuracy with *sigmoid* activation function and the *Nadam* optimizer with a learning rate of 0.0001. We also conduct experiment with the Survival Analysis datasets to show that our proposed model detection rate outperforms the related works regarding the CAN bus attack detection.

7. In-vehicle CAN Bus Intrusion Detection System using 1D CNN Deep Learning Approach

7.1 Introduction

The rapid growth of connected technologies, such as the Internet of Things (IoT), and the ubiquitous nature of the Internet have made life more convenient for human beings. The rise of that social convenience is accompanied by incessant efforts of miscreants to create new tools, techniques and tactics to destabilize the comfort of the dwellers of the connected world. Indeed, we are experiencing a rapid growth of technological weapons of mass destruction that can be used by anyone with elementary computer literacy. Cars are getting increasingly connected; thus, they are becoming a new playground for hackers.

The modern car is a complex piece of technology that contains millions of lines of code that facilitate the interconnection between the electronic control units (ECUs). The Controller Area Network (CAN) bus protocol is the central communication system between ECUs. CAN ID is the destination address of the designated ECU. Because the maximum length of payload is only 8 bytes, there is no encryption or authentication mechanism. This latter state is the origin of all the security issues of the modern car. In fact, the absence of basic security mechanisms makes it easy for attackers to perpetrate several kinds of critical attacks: Denial of Service (DoS), Fuzzing, Malfunction, etc. Thus, it is essential to develop a robust methodology to detect and mitigate such kinds of critical attacks [62].

We believe that, with the aforementioned absence of encryption and authentication mechanisms, an intrusion detection system (IDS) is the most adequate solution to protect CAN bus systems of modern cars [44, 24]. Fig. 17 shows the typical IDS regarding In-vehicle network attack security. An attacker may launch an attack into the real car by using external connections such as an OBD-II diagnostics port, a telematics unit, or in-vehicle infotainment (IVI). An IDS can be placed between the external connection and the CAN bus. In case the attacker tries to inject malicious traffic into the CAN bus system, the IDS will filter the malicious traffic and provide the alert messages.

Researchers in academia and industry professionals have been working diligently to detect and mitigate the CAN bus network attacks. They proposed several methodologies which appear to be lacking adequate and effective attack detection and mitigation technologies. Most of the IDS are inefficient due to the high false positive and false negative alarms. A reason that may explain this fact is that most researchers rely on simulated data to test their proposals. We contend that it is essential to experiment and validate the effectiveness of an IDS with datasets of real cars CAN bus network attacks.

Deep learning has proven to be the right tool for modern IDS. Hence, as a solution to the issue we tackle in this chapter, we propose a Convolutional Neural Network (CNN)-based IDS for in-vehicle CAN bus network attacks detection. For the experiment, we developed an in-vehicle network attacks dataset, which consists of DoS, Fuzzing, RPM and Gear Spoofing attacks of three different models of cars: Toyota, Subaru and Suzuki.

In this chapter, our major contributions are as follows:

- We develop in-vehicle network attacks (DoS, Fuzzing and Spoofing) datasets by using the CAN messages of three different car models (Toyota, Subaru, and Suzuki).
- We propose an efficient Convolutional Neural Network-based IDS for in-vehicle CAN bus systems for well-known network attacks: DoS, Fuzzing, and Spoofing. We evaluate the performance of the IDS regarding in-vehicle network attacks.
- Regarding the improvement of the detection rate, we select the best hyper-parameter values to develop an effective CAN bus IDS based on CNN.

7.2 Related Works

In this section, we discuss the related work regarding network anomaly detection of in-vehicle systems. Researchers have applied a variety of anomaly detection techniques. Several techniques were used for CAN bus anomaly detection, e.g., Statistical, Frequency-based, Hidden Markov Model (HMM), etc.

Song et al. [62] proposed a lightweight IDS for in-vehicle networks based on a time interval analysis (TIA) of CAN's messages. Hyunsang et al. [43] developed

an IDS by analyzing the request-response messages in the CAN bus, based on an offset ratio and time interval analysis. In [63], the authors proposed a Deep CNN Inception-ResNet Architecture regarding CAN bus attack detection. They experiment with real-car datasets consisting of DoS, Fuzzy, and Spoofing attacks and achieved detection rates of 0.9989 for DoS, 0.9965 for the Fuzzy attack, and 0.9989 and 0.9994 for Gear and RPM spoofing attacks. We applied a 1D CNN model wherein we can directly use the CAN messages as an input into the CNN model without any conversion to the CAN messages. Loukas et al. [46] proposed an LSTM-based intrusion detection system for in-vehicle network systems and compared the performance between several machine learning classifiers. They used robotic vehicles and they conducted their experiment at the cloud-based system by injecting attacks. They achieved overall accuracy at 86.9%, and declared that LSTM is suitable to detect vehicular attacks. Kang et al. [34] developed an unsupervised deep belief network (DBN)-based IDS for detecting the CAN bus network attacks. Koscher et al. are the first to demonstrate that several kinds of wireless attack injections are possible in in-vehicle network systems [42]. In case of compromise of the car's network, several kinds of malfunction injections are possible, and it also becomes possible to take control of the car and to stop the engine, disable the brakes, etc.

In this chapter, we note that in an IDS for in-vehicle networks (CAN bus), deep learning algorithms outperform other methodologies. These methodologies include statistical analysis, frequency-based analysis, and Hidden Markov Model (HMM), etc. Additionally, among the deep learning algorithms, LSTM and CNN provides the best results regarding the Time series and Sequential datasets. Hence, in this chapter, we propose an Deep Learning-based IDS for CAN bus networks that performs better than the related work due to our systematic hyper-parameter values fine-tuning.

7.3 Attacks used in the model

In this section, we note that in IDS for in-vehicle networks (CAN bus), deep learning algorithms outperform other methodologies such as statistical analysis, frequency-based technique, Hidden Markov Model (HMM), etc. CNN provides the best results compared to other deep learning algorithms. Hence, in this

chapter, we propose a CNN-based IDS for in-vehicle CAN bus network attacks detection. We developed the network attack dataset for our experiment, which consist of three different vehicle models to which we inject DoS, Fuzzing and Spoofing attacks, Table 15 shows the dataset details. The attack mentioned above explanation is available in Section 5.1, attack scenarios, and the NAIST In-vehicle CAN attack dataset creation details are available in Section 5.3.

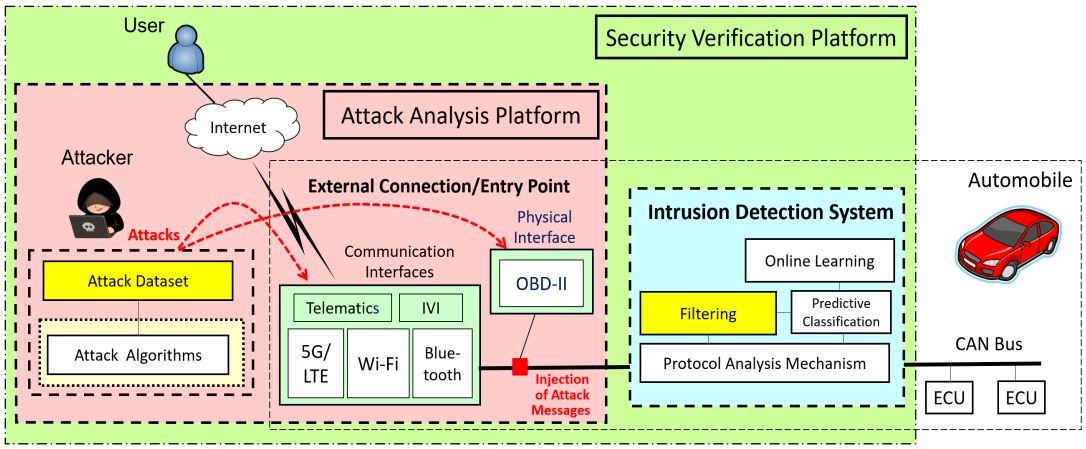


Figure 26. Security Verification Platform

7.4 CNN-based Network Intrusion Detection System

In our proposed CNN model experiment, we used python PyCharm IDE 2020.1 and Keras [2] with TensorFlow as backend. We did our experiment on an Intel Core i9-9000K CPU 3.60 GHz, 64 GB RAM, Windows 10 (64-bit), and NVIDIA GeForce RTX 2080. Table 28 provides the parameter values regarding attacks detection based on CNN multiclass classification models.

Figure 26 depicts the proposed security verification platform's architecture about in-vehicle CAN bus and ECUs. Two modules compose the architecture: the attack analysis platform and the Intrusion Detection System. We depict how an attacker can compromise an In-vehicle CAN bus system through physical and remote access. Moreover, we discussed our security verification proposal. As per the Figure 26 Attack Analysis Platform connection point, an attacker may attempt to compromise the In-vehicle CAN Bus system through OBD-II, Telematics

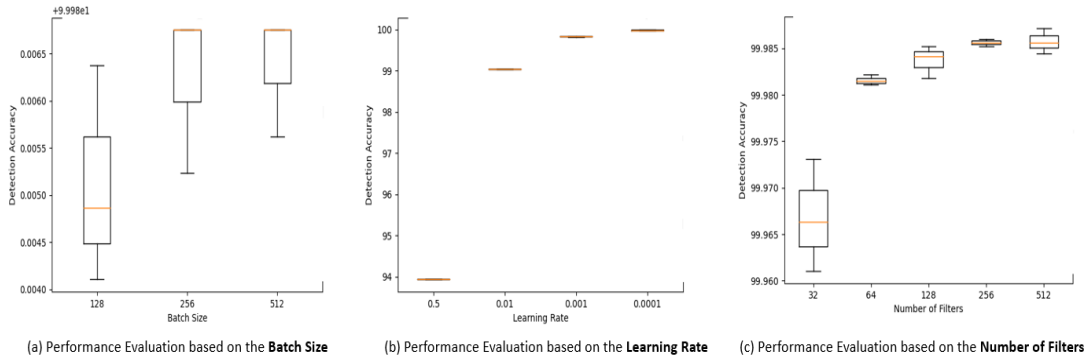
Unit, and IVI communication interfaces. In general, users utilize the Telematics/IVI through LTE, Wi-Fi, Bluetooth, wherein an attacker can inject malicious messages through the technologies mentioned above. When an attacker succeeds in compromising the CAN Bus system, they can take control of a car system and disrupt safe driving. In this research, we contemplate physical access of a car for compromise by using an OBD-II port. In the attack analysis platform, we extract attack-free data from a real-car by OBD-II port. Subsequently, we develop attack algorithms to develop attack(DoS, Fuzzing, and Spoofing) datasets. The IDS module contains the IDS filtering, which is responsible for filtering malicious traffic into the CAN bus message communication.

7.4.1 Dataset Preprocessing

We use NAIST In-vehicle CAN attack dataset; elaborate about the dataset is available in Section 5.3. The NAIST CAN attack dataset is in an hexadecimal format. As per the machine learning requirement, without decoding the data, we converted the hexadecimal format to decimal format and inputed it into the CNN model. We replaced all the blank fields with -1. For our experiment, we considered 10 features and a CAN ID-based IDS. The dataset features contain CAN ID, DLC, Data [D0-D7], and Flag T/R column. CAN ID is an identifier of the CAN messages. DLC data bytes is from 0-8. The data field contains 64 bits, and we position each byte in specific columns such as D0-D7. We denominate R as benign and T as attack class.

For a multiclass classification perspective, we concatenate all the three attack classes for each car model. The results of the concatenation are visible in Table 15.

As per Table 15, the Toyota dataset contains 74.73% of benign class elements and 20.23%, 4.43%, 0.31% and 0.29% of DoS, Fuzzing, RPM and Gear Spoofing elements, respectively. The Subaru car's benign elements are 710524 wherein DoS, Fuzzing, RPM, and Gear Spoofing elements are 93843, 20782, 3333, and 2666, respectively. Benign elements represent 94.43%, and DoS, Fuzzing, RPM, and Gear Malfunction elements are 4.70%, 0.55%, 0.17%, and 0.15%, respectively, in the Suzuki. We noted that the number of benign class elements is higher compared to the attack class elements.



(a) Performance Evaluation based on the **Batch Size** (b) Performance Evaluation based on the **Learning Rate** (c) Performance Evaluation based on the **Number of Filters**

Figure 27. Performance Evaluation (50 epoch) of Suzuki model dataset based on the Batch Size, Learning Rate and Number of Filter maps

7.4.2 Applying the Convolutional Neural Network (CNN) Model

Convolutional Neural Networks (CNN) are one of the most popular deep learning architectures. The popularity of CNN has been exponentially growing since 2012 after the debut of AlexNet. CNN has been successfully applied to many real-time applications. We can use the 1D CNN model regarding time series dataset, that is why we consider the 1D CNN model for our experiment. We split the dataset into training and testing [14]. We trained our CNN classifier, we used 20% for testing. Table 28 shows the CNN parameter values which are used for our experiment.

We experiment with a CNN model to which we apply binary and multiclass classifications. We use a dropout of 0.1 before the pooling layer to avoid overfitting.

We use the Keras [2] API for our experiment. Regarding binary classification, we use the *Adam* optimizer with a learning rate of 0.0001. We employed *binary_crossentropy* as the loss function and *tanh* as an input activation function *sigmoid*, as an output activation function. In Convolutional Neural Network, *Filter* is one of the key parameters which significantly impacts the performance of the IDS. Fig. 27 shows that batch size 256 and 512, learning rate 0.0001, 256 and 512 filter sizes are effective compared to lower filter sizes because they provide effective accuracy and low variance. We use a single convolutional layer with a 512 filter for the binary classification model, and we used two layers with the 512

filters for each layer for the multiclass classification model. In the proposed multiclass CNN model experiment, we use the *Nadam* optimizer with an optimizer's learning rate of 0.0001, and the remaining parameters of the optimizer were used with their default values. We used *categorical_crossentropy* as the loss function. We set the number of iterations to 200 epochs. We accomplished a validation by the *fit()* function by using validation data. After training and testing, we calculated the accuracy and loss based on a number of correctly classified instances. We need to carefully choose the hyper-parameter values due to the fact that it significantly impacts the detection accuracy and detection rate.

Table 28. Parameter values for Multiclass Classification

Parameters	**Value**
Activation Function Input	sigmoid
Epoch	200
Dropout	0.1
Activation Function Output	softmax
Optimizer	Nadam
Batch Size	256
Learning Rate	0.0001
Loss Function	categorical_crossentropy
Encoder	Label Encoder

7.5 Experiment Results and Performance Evaluation

In machine learning, performance measurement is a key fact to evaluate the model's strength. We consider the detection accuracy, detection rate, false positive rate (FPR), false negative rate (FNR) and F1 score to evaluate the model's performance regarding CAN bus IDS. In the case of the imbalanced dataset, the F1 score is an indispensable factor in evaluating the performance of the machine learning algorithms. We cannot consider the accuracy by only regarding the imbalanced dataset [33].

7.5.1 CNN Binary and Multiclass Classification Experiment Results

In Table 29, we observe that CNN can classify DoS and Spoofing attacks with 100% accuracy. It is difficult to detect the fuzzing attack with higher accuracy because the fuzzing attack traffic almost seems like the legitimate traffic, it is challenging to differentiate them. Our classifier detects the Fuzzing attacks with high accuracy for all three car models, which are 99.92%, 99.96%, and 99.94% for Toyota, Subaru, and Suzuki, respectively.

Table 29. CNN Binary Classification Results

Model	Attack	Acc	Recall	F1	FPR	FNR
Toyota	DoS	100%	1.0000	1.0000	0.0000	0.0000
	Fuzzing	**99.92%**	**0.9971**	0.9980	0.0003	0.0029
	RPM	100%	1.0000	1.0000	0.0000	0.0000
	Gear	100%	1.0000	1.0000	0.0000	0.0000
Subaru	DoS	100%	1.0000	1.0000	0.0000	0.0000
	Fuzzing	**99.96%**	**0.9963**	0.9981	0.0000	0.0037
	RPM	100%	1.0000	1.0000	0.0000	0.0000
	Gear	100%	1.0000	1.0000	0.0000	0.0000
Suzuki	DoS	100%	1.0000	1.0000	0.0000	0.0000
	Fuzzing	**99.94%**	**0.9718**	0.9857	0.0000	0.0282
	RPM	100%	1.0000	1.0000	0.0000	0.0000
	Gear	100%	1.0000	1.0000	0.0000	0.0000

Gradient descent is one of the most popular and often widely used as a black-box optimizer algorithm to optimize neural networks. To optimize the machine learning algorithms and optimize the neural networks, it is preferred and popular to use the Gradient descent optimization algorithms [55]. We observe that *Adam* and *Nadam* are suitable optimizers regarding binary and multiclass classifications, respectively, for CAN bus network attack detection. We achieve the best detection accuracy with both optimizers.

The optimizer's learning rate largely impacts the model to achieve reasonable detection accuracy. It is recommended to use the optimizer with default values except for changes in the learning rate. It is better to use a lower learning rate

Table 30. CNN Multicass Classification Results

Model	Attack	Acc	Recall	F1	FPR	FNR
Toyota	Benign	**99.9856%**	1.0000	0.9999	0.0006	0.0000
	DoS		1.0000	1.0000	0.0000	0.0000
	Fuzzing		**0.9967**	0.9984	0.0000	0.0033
	RPM		1.0000	1.0000	0.0000	0.0000
	Gear		1.0000	1.0000	0.0000	0.0000
	Avg		0.9993	0.9997	0.0001	0.0007
Subaru	Benign	**99.9910%**	1.0000	0.9999	0.0006	0.0000
	DoS		1.0000	1.0000	0.0000	0.0000
	Fuzzing		**0.9964**	0.9982	0.0000	0.0036
	RPM		1.0000	1.0000	0.0000	0.0000
	Gear		1.0000	1.0000	0.0000	0.0000
	Avg		0.9993	0.9996	0.0001	0.0007
Suzuki	Benign	**99.9897%**	1.0000	0.9999	0.0019	0.0000
	DoS		1.0000	1.0000	0.0000	0.0000
	Fuzzing		**0.9813**	0.9906	0.0000	0.0187
	RPM		1.0000	1.0000	0.0000	0.0000
	Gear		1.0000	1.0000	0.0000	0.0000
	Avg		0.9963	0.9981	0.0004	0.0037

instead of using the large learning rate to avoid the risk of an overpass of the global minima, but a small learning rate implies longer duration to train the model.

As per Table 30, we observe that our CNN model can classify most of the attacks with effective detection accuracy for all three models of the car attack dataset, which is 99.9856%, 99.9910% and 99.9897% for the Toyota, Subaru and Suzuki, respectively. In the Suzuki dataset, there are only 0.55% fuzzing elements, and we achieve the detection rate of 0.98, which is lower compared to the Toyota and Subaru fuzzing detection rates. From the experiment results we observe that DoS and Spoofing attack detection accuracy is 100%. Our assumption is that the DoS and Spoofing attacks contain a single CAN ID and the attacks patterns are

similar; hence, CNN can easily detect those attacks with high detection accuracy. It is challenging to detect the fuzzing attacks because they contain thousands of random CAN IDs, and the flow of attack patterns mimics the legitimate traffic. Our proposed CNN model can effectively detect the Fuzzing attacks with low false-positive and false-negative rates. The false-negative rate may increase a little based on a large testing set, and also it may degrade the accuracy a bit, i.e., using the 30% testing data set.

7.6 Discussions

An effective and robust intrusion detection system regarding in-vehicle network attack detection is essential for safely driving the modern car. Due to the lack of security mechanisms in the CAN bus system, malicious users can easily inject critical attacks into the CAN bus network; thus, making safe driving concerning. An effective IDS can protect the modern vehicle from critical network attacks. The CAN bus system broadcasts the messages to all the ECUs; hence, it is attractive to attackers for injecting attacks, into the CAN bus network, that can create malfunctioning. In case attackers succeed to inject attacks and compromise the car, then they can stop the engine, disable the brakes and may be able to take control of the entire car system.

We proceeded with a systematic experiment and fine-tuned the model by changing the layer size, filters, optimizer, learning rate, activation and loss function, etc. and selected the best hyperparameter values regarding the improvement of the detection accuracy. Our experiment results show that, to develop an effective IDS, we need to choose the right hyper-parameter values.

Our proposed CNN-based IDS is effective in detecting the CAN bus network attacks. We can detect the attacks without decoding the CAN packets. The dataset contains three different car models Toyota, Subaru, and Suzuki, and each model contains DoS, Fuzzing, RPM, and Gear Spoofing attack. Our CNN model effectively classified all those attacks for individual vehicle models. We achieved a detection accuracy of 99.99% for all the three models of vehicles. In the future, we will consider implementing our classifier into a real car system and evaluate the performance to detect the unknown attacks.

7.7 Chapter Summary

In this chapter, we propose a robust intrusion detection system regarding in-vehicle CAN bus network system based on 1D CNN deep learning approach. For efficiency reason, we develop an In-vehicle CAN Bus attack dataset from three distinct car models: Toyota, Subaru and Suzuki. As per our experiment results, we achieve a high attack detection rate regarding CAN bus network attack detection. We consider Fuzzing as the most critical attack in the in-vehicle system, and it is difficult to detect it because it is similar to the legitimate traffic into the CAN bus network. Our model classifies the Fuzzing attack with high detection accuracy. The proposed 1D CNN model proved that deep learning-based intrusion detection systems are more effective and robust than other methodologies.

8. Discussion and Future Works

In this thesis, our first aim is to investigate several kinds of critical cyber attacks such as Brute Force, DoS, DDoS, and develop an effective deep learning-based Intrusion Detection System. Then, we investigate further on how to optimize the deep learning model to develop a robust IDS concerning the detection of cyber attacks. Our key objective (RO 1.2) is to transfer and adapt (RO 1.1) IDS solutions into the automotive system to improve the vehicle safety and detect a few critical CAN network attacks.

Regarding the improvement of automotive safety, our key aim is to generate a real-time In-vehicle CAN bus attack dataset to develop a robust IDS to detect the critical In-vehicle CAN bus network attacks. We develop the CAN attack dataset by extracting the attack-free traffic from real cars: Toyota, Subaru and Suzuki. We generate attack datasets by injecting three kinds of attacks –DoS, Fuzzing, and Spoofing– into the attack-free dataset. These types of attacks are considered the most prominent in the CAN bus system, in terms of damage capabilities. For the first experiment, we develop attack dataset from the extracted attack-free data from a Toyota car (Section 5.2) and, for next, we extend the dataset development by adding two other real car models, Subaru and Suzuki, along with the Toyota model (Section 5.3). Additionally, our key objective was to try to mimic real-life scenarios to analyze the imbalance impacts in our experiments and evaluate the performance of the IDS. The connected cars market is rapidly growing; hence, the interest of hackers is proportionally growing. CAN bus data security is of utmost importance regarding safe driving.

Consequently, we propose long short-term memory (LSTM)-based Intrusion Detection System (IDS) for in-vehicle CAN bus network attack by using the NAIST CAN attack dataset (Chapter 6). As per the experiment results, LSTM-based IDS is effective in detecting the In-vehicle CAN bus network attacks. We propose an effective pre-processing method, and our proposed LSTM model can classify benign and attack classes with a high accuracy. Finally, we extend our dataset and develop the In-vehicle CAN bus attack dataset by using aforementioned three real model car. For the effectiveness, we also further investigate by applying another deep learning model 1D Convolutional Neural Network (1D CNN) for CAN bus attack detection (Chapter 7). As per our experiment re-

sults,the proposed 1D CNN model proves that deep learning-based intrusion detection systems are more effective and robust than other methodologies.

Although the proliferation of intelligent hacking tools, techniques, and tactics has allowed anyone with basic computer literacy to be able to attack automotive systems, our proposed solutions will provide effective solutions and significantly impact the resilience of connected and modern automotive systems.

8.1 Limitations

The major limitation of our model resides in the fact that we experiment in offline mode with labeled datasets; we are concerned regarding the performance of the IDS on online mode and also about the IDS effectiveness regarding unknown attack detection. Additionally, we have not defined how to recover the vehicle system subsequent to injecting the attacks, i.e., the CAN bus system must be available even after our attack injections. Another key challenge is about how to embed the deep learning-based IDS with the CAN bus system due to limited processing power. In fact, a real-world dataset is quite essential to evaluate a model's performance and develop robust solutions regarding automotive security. In this research, we consider developing the CAN attack dataset by using the OBD-II port only. We did not study how an attacker can compromise automotive systems through Wi-Fi, Bluetooth, etc.

As we asserted, advanced technologies enlarge the automotive attack surface. Hence, effective security solutions are required regarding the acceptance of autonomous cars. Academia and industry are suppose to pay more attention to improving the security of the modern connected automotive systems for safe driving and mitigating cyber attack threats, because there are no standard security solutions for the automotive system. Many stakeholders are involved in the modernization of the automotive systems in which they embedded their advanced technologies, and they did not consider the security features at the time of technology development. The embedded advanced technologies are augmenting the attack surfaces; thus, they constitute a significant threat to automotive systems. It is a significant challenge to secure the whole system at once, because the modern automotive is a system of systems. Moreover, we have to consider the integrated security solutions and standard security framework in the future

for the better adoption of the autonomous car in particular, and digital transformation in general. However, to develop security solutions, real-world datasets are imperative to evaluate the performance.

Researchers whose work revolve around automotive security and deep learning face issues related to the availability of real-time public datasets. However, modern automotive systems are now not only mechanical equipment; they have been modernized with many advanced embedded technologies, millions of code lines are used to facilitate ECUs communication, and concurrently, the rapid modernization of automotive systems is augmenting the attack surface. The industry should pay more attention to automotive security along with the modernization of the automotive system to mitigate car hacking, protect car takeover by hackers, and avoid unexpected accidents. Hence, a real-world dataset is considerably necessary for advanced-level research, developing efficient solutions, and ultimately the resilience of connected cars.

8.2 Future Works: Resilience of Connected Cars

Many stakeholders are involved in the advancement of modern automotive systems. Modern automotive systems consist of many embedded technologies developed by various vendors. It is challenging to secure the automotive system as a whole because of large attack surfaces. Industry and academia need to pay more attention to developing sophisticated methodologies to improve automotive safety and mitigate cyber attacks. Numerous challenges persist in securing the most advanced embedded technologies and protecting the automotive cyber-physical system. Researchers are required to take into account advanced security measures to protect the automotive systems. To ensure safe driving, the most important fact to consider is to effectively detect the critical cyber attacks and develop a protection mechanism to drive the vehicle without interruption after attacks' injection. Our aim is to design and develop new security mechanisms for the modern automotive system or autonomous car that are compliant with the CIA (Confidentiality, integrity, and availability) triad.

8.2.1 Automotive Attacks Investigation

We consider a few critical In-vehicle CAN bus network attacks in this research. The rapid transformation of the modern car increases the attack surfaces, and an attacker can inject attacks physically or through remote access. The present work is based on the physical interface access to investigate the In-vehicle CAN bus attacks. We will moreover explore different attack surfaces; also, we will contemplate some distinct attack patterns.We will also further intensify the detection of unknown attacks. Additionally, we will analyze unsupervised deep learning approaches to develop a robust IDS about automotive security.

8.2.2 IDS embedding into a real-car

In this study, we develop an effective IDS regarding In-vehicle CAN Bus network attack detection, but it's challenging to embed the deep learning-based IDS into a real-car. We will further analyze and design how to embed an IDS into a real-car in our future work. An IDS can be placed as a gateway of the CAN bus system network or placed in between the ECUs. A more detailed investigation is required for the effective placing of the IDS. Nevertheless, it's more challenging and concerning to embed the IDS because of the automotive system's limited processing power. Addressing these challenges to embed IDS into a real-car is imperative in future work.

8.2.3 V2X Communication Security

Regarding the advancement of safe driving, the Telematics concept has been introduced. The Vehicle-to-Everything (V2X) concept is to communicate between vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), as well as any diverse communication systems that affect the vehicle to improve safe driving, mitigate accidents and reduce traffic, etc. [74, 5, 50, 47]. In V2V, vehicles connect with the other vehicles to exchange data regarding safety to circumvent accidents. Regarding connection and exchange data to other vehicles and Vehicle-to-Infrastructure (V2I), the V2X uses WLAN or cellular networks; thus, it augments the attack surface [74]. The attackers can spoof such kinds of inter-vehicle communication channels or message exchanges; consequently, rendering safe driving questionable.

Moreover, it's imperative to investigate V2X communications, attack possibilities, and detection and protection procedures, and robust methodologies for safe driving.

8.2.4 In-vehicle Infotainment Systems (IVI) Attack Analysis

In-general, the In-vehicle Infotainment system has been developed to improve the driving experience and exchange information for safe driving. Users can connect the IVI to the external networks through Bluetooth, Wi-Fi, GPS. Many advanced instruments, such as V2X, ADAS, Sensors, etc., are connected with the IVI to improve the user's comfort and safe driving [50, 57]. An attacker can easily compromise the connected car with the aforementioned external connection technologies; thus, they can inject malicious malware into the car and disrupt safe driving [30]. A further comprehensive investigation regarding the Wi-Fi-based attack injection and protection mechanism is imperative for the advanced connected car's safe driving.

Users can connect several systems through the Bluetooth networks. In general, Bluetooth is used for short-range wireless connectivity. Users can connect the In-vehicle Infotainment Systems (IVI) through Bluetooth technology. Numerous kinds of security threats concern the Bluetooth connectivity: such as Information leakage, Malware Injection, Denial of Service (DoS) attacks, etc [45]. As a result, an attacker can compromise the car systems through Bluetooth, thus initiating significant threats concerning the car's safe driving. Going forward,investigate and develop an effective solution to prevent modern automotive systems to be compromised through Bluetooth, and further, intricate attack patterns investigation is indispensable for future research directions.

8.2.5 In-vehicle Malware Activities and Analysis

The modern cars are connected; it consists of many advanced pieces of equipment and sensors to improve the connectivity and share information with the outside world but at the same time, that connectivity increases it's security threats. An attacker, with minimum hacking knowledge, can easily inject malicious malware into a vehicle to compromise the ECUs through intelligent attacking tools [79]. In case ECUs are compromised by hackers, the latter can take control of the car and

disrupt the In-vehicle functionalities. A novel solution to defend against malware injection for safe driving is very crucial. Academia and the industry need to pay more attention, address the existing challenges, and find an effective protection system against malware attacks.

In summary, our robust IDS solutions regarding the CAN bus system provide evidence that Artificial Intelligence (deep learning) can play a vital role in the cybersecurity and resilience of connected cars. For future research, deep learning-based solutions will be dominant in terms of automotive security or the other cyber-physical systems domains. Conclusively, we contend that to fulfill the future research directions, develop robust Artificial Intelligence (deep learning)-based solutions concerning connected cars' resilience, a real-world dataset is obligatory. Indeed, academia and industry develop a few datasets, but, because of privacy concerns, they are unable to make them publicly available. Thus, academia and industry must be more collaborative regarding publicly available datasets for academic research.

9. Conclusion

In this research, our primary aim is to develop deep learning-based intrusion detection systems that are effective in Computer and In-vehicle CAN Bus networks. In the beginning, we thoroughly investigate several critical computer network attacks and experiment with the LSTM-based deep learning models to achieve our aims. Our models achieve better performance to detect a few critical computer network attacks. As we said earlier, well-known traditional computer network attacks are being transferred to the In-vehicle CAN Bus network due to the lack of security mechanism. Additionally, there are no standard security solutions regarding automotive cyber attack detection, which significantly undermine safe driving. We also transfer our traditional computer network solutions into the In-vehicle CAN Bus IDS network system to detect sophisticated CAN Bus system attacks. First, we develop In-vehicle CAN Bus attack datasets from a real-car and extend the dataset with three distinct car models. Moreover, we provide an effective pre-processing method, which is utterly essential to develop a robust IDS. Finally, we design and develop In-vehicle CAN bus system IDS based on the LSTM and 1D CNN deep learning approaches. Our efficient IDS solution can mitigate the attacking threat and improve safe driving.

Safety is one of the most significant solicitudes for the adoption of autonomous cars, our methodology can augment the connected car's safety on detecting critical cyber attacks, which can help propel the adoption of autonomous cars. Furthermore, our studies present significant directions regarding effective countermeasures in well-known domains that can be transferred (adapted) to domains that are a new playground for hackers such as smart grid, industrial control systems security, etc. Our investigations provide the approaches and guidelines for adapting our solutions into other challenging domains, i.e., developing less biased attack datasets, the importance of an efficient pre-processing method, and finally developing deep learning-based IDS.

Moreover, deep learning, Artificial Intelligence (AI) is the contemporaneous dominant technology with proven applications in various fields such as image recognition, voice recognition, weather forecasting, market analysis, etc. However, the recent advancements in machine learning and deep learning have resuscitated the hope that we can find solutions to counter the ever-sophisticated

aforementioned attacks, and to realize the resilience of connected cars. There are already trailblazing works in academia, and deep learning approaches have particularly been successfully applied to many application domains. Hence, we contend that it can equally be successfully applied for automotive cyber attacks detection.

Acknowledgments

I would like to express my sincere appreciation to all of the people who are involved in the making of my Ph.D. thesis. First of all, I would like to acknowledge my supervisor, Professor Dr. Youki Kadobayashi, for his guidance and encouragement and for facilitating my collaboration with The University of Tokyo and The Hiroshima City University. I also thank Associate Prof. Dr. Hideya Ochiai (The University of Tokyo) and Associate Prof. Dr. Hiroyuki Inoue for their collective effort to support me since the beginning of my doctoral research journey. I am also grateful to Prof. Keiichi Yasumoto, Prof. Yuichi Hayashi, Associate Prof. Daisuke Miyamoto, Associate Prof. Dr. Yuzo Taenaka and Assistant Prof. Dr. Doudou Fall for their advice and comments. I want to express my gratitude to the secretaries and other laboratory members for their support and cooperation regarding academic as well as daily life matters. I would like to thank my wife and two daughters for their tremendous support during my Doctoral Research. I am also thankful to the Doctoral fellowship Committee members - Ministry of Posts, Telecommunication and Information Technology, ICT Division, Bangladesh.

I am remarkably grateful to the **Industrial Cyber Security Center of Excellence (ICS-CoE)** for providing me Research Assistantship during my entire Ph.D. research. I am grateful to the following funding organizations who supported my research: ICS-CoE, JST CREST, JSPS KAKENHI, MEXT Japan, and **Ministry of Posts, Telecommunication and Information Technology, ICT Division, Bangladesh**.

References

[1] *colah's blog, Understanding LSTM Networks*, 2020 (accessed August 15, 2020). http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[2] *Keras: The Python Deep Learning library*, 2020 (accessed January 10, 2020). https://keras.io/.

[3] *Scikit-Learn Project, "Receiver Operating Characteristic (ROC)*, 2020 (accessed March 03, 2020). https://scikit-learn.org/stable/auto$_e$*xamples*/*model*$_s$*election*/*plot*$_r$*oc.htm*.

[4] *Cyber-Physical Systems*, 2020 (accessed Nov 17, 2020). https://cyberphysicalsystems.org.

[5] $V2X_Autotalks, [Online; accessed 21 - November - 2020]. https://www.auto-talks.com/technology/v2x-wiki/.$

[6] Mouhammd Alkasassbeh, Ghazi Al-Naymat, Ahmad Hassanat, and Mohammad Almseidin. Detecting distributed denial of service attacks using data mining techniques. *International Journal of Advanced Computer Science and Applications*, 7(1):436–445, 2016.

[7] Md Zahangir Alom, VenkataRamesh Bontupalli, and Tarek M Taha. Intrusion detection using deep belief networks. In *2015 National Aerospace and Electronics Conference (NAECON)*, pages 339–344. IEEE, 2015.

[8] Khaled Alrawashdeh and Carla Purdy. Toward an online anomaly intrusion detection system based on deep learning. In *2016 15th IEEE international conference on machine learning and applications (ICMLA)*, pages 195–200. IEEE, 2016.

[9] Sara A Althubiti, Eric Marcell Jones, and Kaushik Roy. Lstm for anomaly-based network intrusion detection. In *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–3. IEEE, 2018.

[10] Sara A Althubiti, Eric Marcell Jones, and Kaushik Roy. Lstm for anomaly-based network intrusion detection. In *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–3. IEEE, 2018.

[11] Jason Brownlee. *How to Choose Loss Functions When Training Deep Learning Neural Networks*, 2020 (accessed April 024, 2020). https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/.

[12] Jason Brownlee. *Stacked Long Short-Term Memory Networks*, 2020 (accessed Feb 04, 2020). https://machinelearningmastery.com/stacked-long-short-term-memory-networks/.

[13] Jason Brownlee. *How to Develop LSTM Models for Time Series Forecasting*, 2020 (accessed Feb 05, 2020). https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/.

[14] Jason Brownlee. *1D Convolutional Neural Network Models for Human Activity Recognition*, 2020 (accessed March 02, 2020). https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/.

[15] Jason Brownlee. *How to Configure the Number of Layers and Nodes in a Neural Network*, 2020 (accessed March 02, 2020). https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/.

[16] David Freeman Clarence Chio. *Machine Learning and Security*. O'Reilly, 2018.

[17] Christian Darken, Joseph Chang, John Moody, et al. Learning rate schedules for faster stochastic gradient search. In *Neural networks for signal processing*, volume 2, 1992.

[18] Martin Drašar. Protocol-independent detection of dictionary attacks. In *Meeting of the European Network of Universities and Companies in Information and Communication Engineering*, pages 304–309. Springer, 2013.

[19] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[20] Asmaa Elsaeidy, Kumudu S Munasinghe, Dharmendra Sharma, and Abbas Jamalipour. Intrusion detection in smart cities using restricted boltzmann machines. *Journal of Network and Computer Applications*, 135:76–83, 2019.

[21] Canadian Institute for Cybersecurity. *CIC DoS dataset (2017)*, 2020 (accessed April 05, 2020). https://www.unb.ca/cic/datasets/dos-dataset.html.

[22] Rohith Gandhi. *A Look at Gradient Descent and RMSprop Optimizers*, 2019 (accessed November 05, 2019). https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b.

[23] Ni Gao, Ling Gao, Quanli Gao, and Hai Wang. An intrusion detection model based on deep belief networks. In *2014 Second International Conference on Advanced Cloud and Big Data*, pages 247–252. IEEE, 2014.

[24] Roland E Haas, Dietmar PF Möller, Prateek Bansal, Rahul Ghosh, and Srikrishna S Bhat. Intrusion detection in connected cars. In *2017 IEEE International Conference on Electro Information Technology (EIT)*, pages 516–519. IEEE, 2017.

[25] 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.

[26] Laurens Hellemons, Luuk Hendriks, Rick Hofstede, Anna Sperotto, Ramin Sadre, and Aiko Pras. Sshcure: a flow-based ssh intrusion detection system. In *IFIP International Conference on Autonomous Infrastructure, Management and Security*, pages 86–97. Springer, 2012.

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

[28] 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.

[29] Yadigar Imamverdiyev and Fargana Abdullayeva. Deep learning method for denial of service attack detection based on restricted boltzmann machine. *Big Data*, 6(2):159–169, 2018.

[30] Japan IT Security Center, INFORMATION-TECHNOLOGY PROMOTION AGENCY. *Approaches for Vehicle Information Security*, 2020 (accessed Nov 22, 2020). https://www.ipa.go.jp/files/000033402.pdf.

[31] Mobin Javed and Vern Paxson. Detecting stealthy, distributed ssh bruteforcing. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 85–96, 2013.

[32] Michael Jaynes, Ram Dantu, Roland Varriale, and Nathaniel Evans. Automating ecu identification for vehicle security. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 632–635. IEEE, 2016.

[33] Renuka Joshi. *Accuracy, Precision, Recall F1 Score: Interpretation of Performance Measures*, 2019 (accessed April 11, 2019). https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/.

[34] Min-Joo Kang and Je-Won Kang. Intrusion detection system using deep neural network for in-vehicle network security. *PloS one*, 11(6):e0155781, 2016.

[35] Zadid Khan, Mashrur Chowdhury, Mhafuzul Islam, Chin-Ya Huang, and Mizanur Rahman. Long short-term memory neural networks for false information attack detection in software-defined in-vehicle network. *arXiv preprint arXiv:1906.10203*, 2019.

[36] Jihyun Kim, Howon Kim, et al. An effective intrusion detection classifier using long short-term memory with gradient descent optimization. In *2017*

International Conference on Platform Technology and Service (PlatCon), pages 1–6. IEEE, 2017.

[37] Jihyun Kim, Jaehyun Kim, Huong Le Thi Thu, and Howon Kim. Long short term memory recurrent neural network classifier for intrusion detection. In *2016 International Conference on Platform Technology and Service (PlatCon)*, pages 1–5. IEEE, 2016.

[38] Jihyun Kim, Jaehyun Kim, Huong Le Thi Thu, and Howon Kim. Long short term memory recurrent neural network classifier for intrusion detection. In *2016 International Conference on Platform Technology and Service (PlatCon)*, pages 1–5. IEEE, 2016.

[39] Kwangjo Kim, Muhamad Erza Aminanto, and Harry Chandra Tanuwidjaja. *Network Intrusion Detection Using Deep Learning: A Feature Learning Approach*. Springer, 2018.

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

[41] Pierre Kleberger, Tomas Olovsson, and Erland Jonsson. Security aspects of the in-vehicle network in the connected car. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 528–533. IEEE, 2011.

[42] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, et al. Experimental security analysis of a modern automobile. In *2010 IEEE Symposium on Security and Privacy*, pages 447–462. IEEE, 2010.

[43] Hyunsung Lee, Seong Hoon Jeong, and Huy Kang Kim. Otids: A novel intrusion detection system for in-vehicle network by using remote frame. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 57–5709. IEEE, 2017.

[44] Siti-Farhana Lokman, Abu Talib Othman, and Muhammad-Husaini Abu-Bakar. Intrusion detection system for automotive controller area network

(can) bus system: a review. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):184, 2019.

[45] Angela M Lonzetta, Peter Cope, Joseph Campbell, Bassam J Mohd, and Thaier Hayajneh. Security vulnerabilities in bluetooth technology as used in iot. *Journal of Sensor and Actuator Networks*, 7(3):28, 2018.

[46] George Loukas, Tuan Vuong, Ryan Heartfield, Georgia Sakellari, Yongpil Yoon, and Diane Gan. Cloud-based cyber-physical intrusion detection for vehicles using deep learning. *IEEE Access*, 6:3491–3508, 2017.

[47] Radu Popescu-Zeletin Mihai Adrian Rigani, Ilja Radusch. *Vehicular-2-X Communication*. Springer-Verlag Berlin Heidelberg, 2010.

[48] Xiuliang Mo, Pengyuan Chen, Jianing Wang, and Chundong Wang. Anomaly detection of vehicle can network based on message content. In *International Conference on Security and Privacy in New Computing Environments*, pages 96–104. Springer, 2019.

[49] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, 20(4):2923–2960, 2018.

[50] Dietmar P.F. Möller and Roland E. Haas. *Guide to Automotive Connectivity and Cybersecurity*. Springer International Publishing, 2019.

[51] Maryam M Najafabadi, Taghi M Khoshgoftaar, Clifford Kemp, Naeem Seliya, and Richard Zuech. Machine learning for detecting brute force attacks at the network level. In *2014 IEEE International Conference on Bioinformatics and Bioengineering*, pages 379–385. IEEE, 2014.

[52] Benjamin J Radford, Bartley D Richardson, and Shawn E Davis. Sequence aggregation rules for anomaly detection in computer network traffic. *arXiv preprint arXiv:1805.03735*, 2018.

[53] Shiho Kim Rakesh Shrestha. *Automotive Cyber Security*. Springer Singapore, 2020.

[54] Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho. A survey of network-based intrusion detection data sets. *Computers & Security*, 86:147–167, 2019.

[55] Sebastian Ruder. *An overview of gradient descent optimization algorithms*, 2019 (accessed November 02, 2019). https://ruder.io/optimizing-gradient-descent/.

[56] Akihiro Satoh, Yutaka Nakamura, and Takeshi Ikenaga. Ssh dictionary attack detection based on flow analysis. In *2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet*, pages 51–59. IEEE, 2012.

[57] Anshul Saxena. *Everything You Need to Know About In-Vehicle Infotainment Systems*, 2020 (accessed Nov 18, 2020). https://www.einfochips.com/blog/everything-you-need-to-know-about-in-vehicle-infotainment-system.

[58] 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.

[59] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, pages 108–116, 2018.

[60] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. A deep learning approach to network intrusion detection. *IEEE transactions on emerging topics in computational intelligence*, 2(1):41–50, 2018.

[61] Koo Ping Shung. *Accuracy, Precision, Recall or F1?*, 2019 (accessed November 01, 2019). https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

[62] Hyun Min Song, Ha Rang Kim, and Huy Kang Kim. Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network. In *2016 international conference on information networking (ICOIN)*, pages 63–68. IEEE, 2016.

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

[64] Deris Stiawan, Mohd Idris, Reza Firsandaya Malik, Siti Nurmaini, Nizar Alsharif, Rahmat Budiarto, et al. Investigating brute force attack patterns in iot network. *Journal of Electrical and Computer Engineering*, 2019, 2019.

[65] Intrepid Control Systems. *Vehicle Spy Enterprise*, 2020 (accessed March 06, 2020). https://intrepidcs.com/products/software/vehicle-spy/.

[66] 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.

[67] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*, 2012.

[68] Hung Nguyen Viet, Quan Nguyen Van, Linh Le Thi Trang, and Shone Nathan. Using deep learning model for network scanning detection. In *Proceedings of the 4th International Conference on Frontiers of Educational Technologies*, pages 117–121, 2018.

[69] Jan Vykopal, Martin Drašar, Philipp Winter, et al. *Flow-based brute-force attack detection*. Fraunhofer Verlag, 2013.

[70] Jan Vykopal, Tomas Plesnik, and Pavel Minarik. Network-based dictionary attack detection. In *2009 international conference on future networks*, pages 23–27. IEEE, 2009.

[71] Wikipedia contributors. Intrusion detection system — Wikipedia, the free encyclopedia, 2020. https://en.wikipedia.org/w/index.php?title=Intrusion$_d etection_s ystemoldid$ = 997247832.

[72] Wikipedia contributors. On-board diagnostics — Wikipedia, the free encyclopedia, 2020. https://en.wikipedia.org/w/index.php?title=On-board$_d$*iagnosticsoldid* = 993657894.

[73] Wikipedia contributors. Precision and recall — Wikipedia, the free encyclopedia, 2020. https://en.wikipedia.org/w/index.php?title=Precision$_a$*nd$_r$ecalloldid* = 995861774.

[74] Wikipedia contributors. Vehicle-to-everything — Wikipedia, the free encyclopedia, 2020. https://en.wikipedia.org/w/index.php?title=Vehicle-to-everythingoldid=984798501.

[75] Samuel Woo, Hyo Jin Jo, and Dong Hoon Lee. A practical wireless attack on the connected car and security protocol for in-vehicle can. *IEEE Transactions on intelligent transportation systems*, 16(2):993–1006, 2014.

[76] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[77] Xiaoyong Yuan, Chuanhuang Li, and Xiaolin Li. Deepdefense: identifying ddos attack via deep learning. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8. IEEE, 2017.

[78] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[79] Tao Zhang, Helder Antunes, and Siddhartha Aggarwal. Defending connected vehicles against malware: Challenges and a solution framework. *IEEE Internet of Things journal*, 1(1):10–21, 2014.

[80] Guangzhen Zhao, Cuixiao Zhang, and Lijuan Zheng. Intrusion detection using deep belief network and probabilistic neural network. In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, volume 1, pages 639–642. IEEE, 2017.

Publication List

Journals

[1] **Hossain, Md Delwar**, Hiroyuki Inoue, Hideya Ochiai, Doudou Fall, and Youki Kadobayashi. "LSTM-Based Intrusion Detection System for In-Vehicle Can Bus Communications." IEEE Access 8 (2020): 185489-185502.

International Conferences

[1] **Hossain, Md Delwar**, Hiroyuki Inoue, Hideya Ochiai, Doudou Fall, and Youki Kadobayashi. "An Effective In-Vehicle CAN Bus Intrusion Detection System Using CNN Deep Learning Approach." In GLOBECOM 2020-2020 IEEE Global Communications Conference, pp. 1-6. IEEE, 2020.

[2] **Hossain, Md Delwar**, 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), pp. 10-17. IEEE, 2020.

[3] **Hossain, Md Delwar**, Hideya Ochiai, Doudou Fall, and Youki Kadobayashi. "LSTM-based Network Attack Detection: Performance Comparison by Hyper-parameter Values Tuning." In 2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), pp. 62-69. IEEE, 2020.

[4] **Hossain, Md Delwar**, Hideya Ochiai, Doudou Fall, and Youki Kadobayashi. "SSH and FTP brute-force Attacks Detection in Computer Networks: LSTM and Machine Learning Approaches." In 2020 5th International Conference on Computer and Communication Systems (ICCCS), pp. 491-497. IEEE, 2020.