Received September 8, 2020, accepted September 21, 2020, date of publication October 7, 2020, date of current version October 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3029307

LSTM-Based Intrusion Detection System for In-Vehicle Can Bus Communications

MD DELWAR HOSSAIN[1], **(Graduate Student Member, IEEE),**
HIROYUKI INOUE[2], **(Member, IEEE), HIDEYA OCHIAI**[3], **(Member, IEEE),**
DOUDOU FALL[1], **AND YOUKI KADOBAYASHI**[1], **(Member, IEEE)**
[1]Division of Information Science, Graduate School of Science and Technology, Nara Institute of Science and Technology, Ikoma 630-0192, Japan
[2]Graduate School of Information Sciences, Hiroshima City University, Hiroshima 731-3194, Japan
[3]Graduate School of Information Science, The University of Tokyo, Tokyo 113-8654, Japan

Corresponding author: Md Delwar Hossain (hossain.md_delwar.hi5@is.naist.jp)

This work was supported in part by the ICS-CoE Core Human Resources Development Program, in part by the JST CREST under Grant JPMJCR1783, in part by the JSPS KAKENHI, Japan, under Grant JP18K11299, in part by the MEXT Scholarship, and in part by the Ministry of Posts, Telecommunication, and Information Technology, ICT Division, Bangladesh, under Grant 56.00.0000.028.33.002.19.8.

ABSTRACT The modern automobile is a complex piece of technology that uses the Controller Area Network (CAN) bus system as a central system for managing the communication between the electronic control units (ECUs). Despite its central importance, the CAN bus system does not support authentication and authorization mechanisms, i.e., CAN messages are broadcast without basic security features. As a result, it is easy for attackers to launch attacks at the CAN bus network system. Attackers can compromise the CAN bus system in several ways including Denial of Service (DoS), Fuzzing and Spoofing attacks. It is imperative to devise methodologies to protect modern cars against the aforementioned attacks. In this paper, we propose a Long Short-Term Memory (LSTM)-based Intrusion Detection System (IDS) to detect and mitigate the CAN bus network attacks. We generate our own dataset by first extracting attack-free data from our experimental car and by injecting attacks into the latter and collecting the dataset. We use the dataset for testing and training our model. With our selected hyper-parameter values, our results demonstrate that our classifier is efficient in detecting the CAN bus network attacks, we achieved an overall detection accuracy of 99.995%. We also compare the proposed LSTM method with the Survival Analysis for automobile IDS dataset which is developed by the Hacking and Countermeasure Research Lab, Korea. Our proposed LSTM model achieves a higher detection rate than the Survival Analysis method.

INDEX TERMS Modern car security, controller area network, deep learning, LSTM, intrusion detection system.

I. INTRODUCTION

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. 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.

The associate editor coordinating the review of this manuscript and approving it for publication was Tariq Umer.

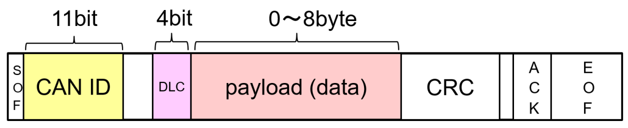


FIGURE 1. CAN message format in 11bit mode with DLC=8. There are no security features implemented in this protocol.

- **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.

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 [1], [2]. The main security issues of the CAN bus arise because it broadcasts all the ECUs' messages without encryption nor authentication [3]. 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.* [4] demonstrated that it is possible to compromise the CAN bus system and ECUs by investigating wireless attacks into the vehicle system. 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 paper, as an extension of our previous work [5] 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 [2]. 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., [6].

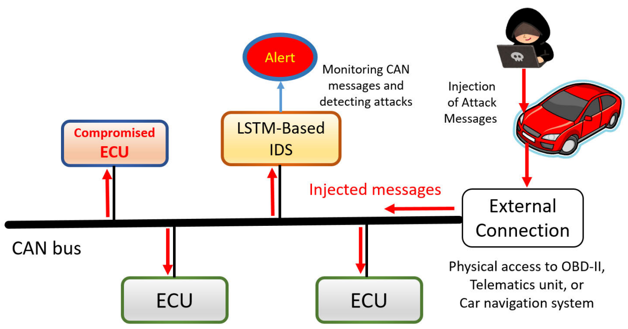

Alert
Monitoring CAN messages and detecting attacks
Injection of Attack Messages
Compromised ECU
LSTM-Based IDS
Injected messages
External Connection
CAN bus
ECU
ECU
Physical access to OBD-II, Telematics unit, or Car navigation system

FIGURE 2. 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.

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 [7], [8]. A signature-based IDS is unable to detect unknown attacks, whereas an anomaly-based IDS is capable of detecting unknown and malfunction attacks.

Fig. 2 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.

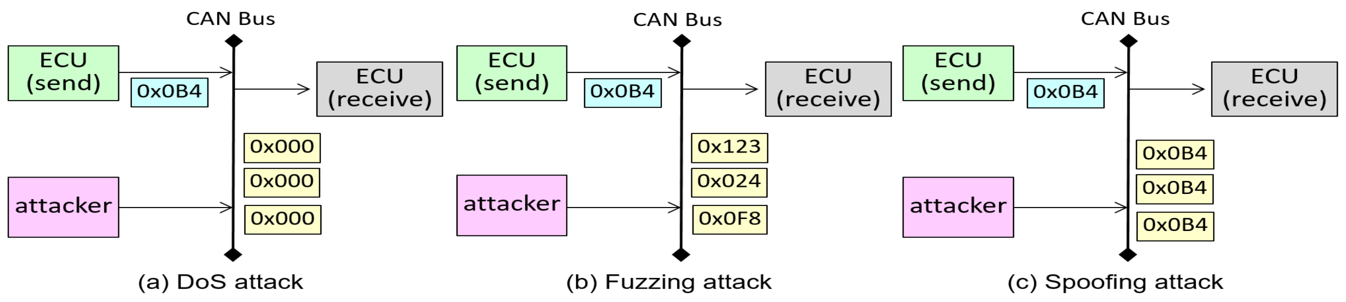


FIGURE 3. **Attack scenarios assumed in this paper. (a) DoS attack – an attacker floods messages to the CAN bus. (b) Fuzzing attack – an attacker injects random CAN messages for changing the IDs, payload length. (c) Spoofing attack – an attacker generates fake messages that deceive the receiver's ECUs.**

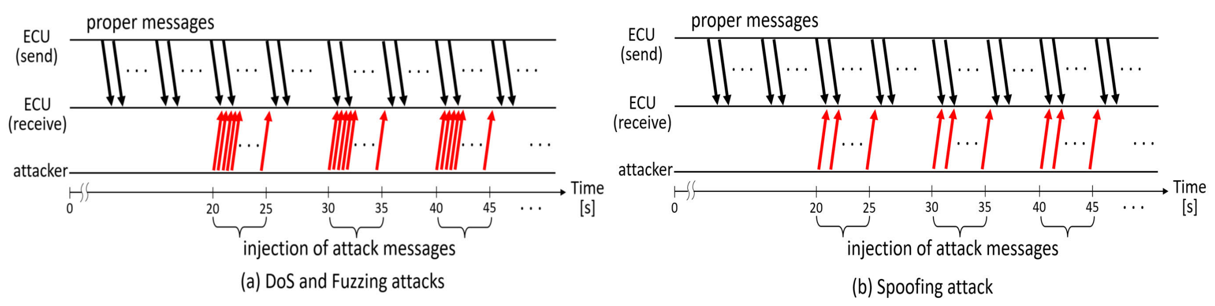


FIGURE 4. **Injection of messages with timing regarding dos, fuzzing and spoofing attacks.**

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 [2] 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.

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.

The remainder of this paper is organized as follows. Section II discusses the related works. In Section III, we describe the overview of the dataset we use for deep learning before proceeding to Section IV to explain our proposal. Section V contains the experimental results and performance evaluations. In Section VI, we provide discussions and future works, and Section VII concludes this manuscript.

II. 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 [4]. 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 [9]. They also argued about IDS and architectural security features. Loukas *et al.* [10] 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%. Seo *et al.* [1] 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 [2]. They extracted attack data from a real car, and they also injected attacks to generate: Flooding, Fuzzy, and Malfunction attack datasets. Kang *et al.* [11] 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 [8], 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 [12] 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.* [13] proposed an IDS for in-vehicle networks based on a time interval analysis (TIA) of the CAN messages.

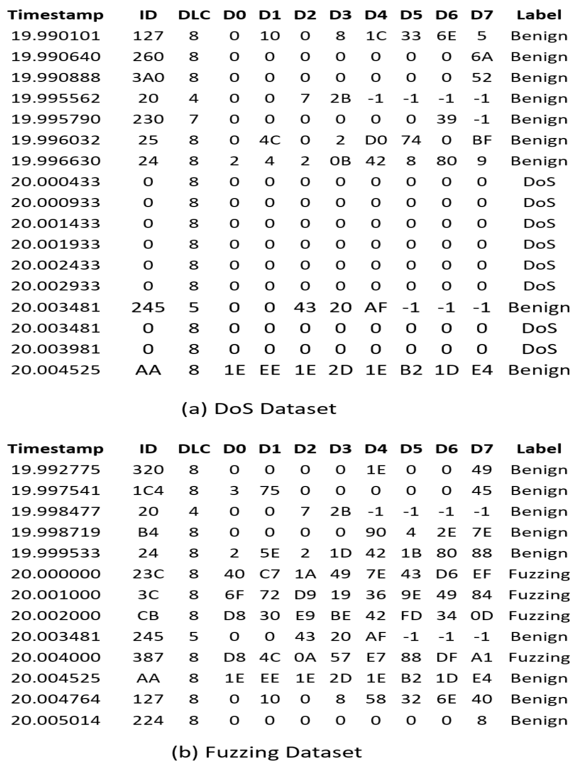
Lee *et al.* [14] developed an IDS by analyzing the request-response message in the CAN bus, based on an offset ratio and time interval analysis. Khan *et al.* [15] Khan *et al.* investigated SDN-based false data injection into the brake-related ECUs. They developed false information attack dataset and applied LSTM to detect the attack, and they achieved a detection rate of 87%. Woo *et al.* [16] 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.* [17] 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.

In this section, 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 provides the best results. Hence, in this paper, we propose an LSTM-based IDS for CAN bus networks that performs better than the related work due to our systematic hyper-parameter values fine-tuning.

III. ATTACKS USED IN THE MODEL

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

DoS Attack: The key objective of a DoS attack in the CAN bus is to interrupt or disable the services between the ECUs. During a DoS attack, attackers continuously send arbitrary messages with high priority bits in the CAN ID. Thus, high frequency and high priority CAN messages occupy the CAN bus network system. As a result, legitimate messages transmissions between the ECUs are obstructed.



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
20.003481	0	8	0	0	0	0	0	0	0	0	DoS
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 5. **Examples of NAIST CAN attack dataset.**

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., [14]. 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: In a Spoofing attack, an intruder targets specific CAN IDs to inject modified messages; thus, the ECUs get biased. Consequently, it becomes challenging to identify legitimate messages, and the system may start to malfunction.

A. DATASET GENERATION

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 for a duration of 120 seconds. We design three attack scenarios –DoS, Fuzzing, and Spoofing– by creating the attack datasets based on the real car messages. Fig. 5 depicts an example of the NAIST CAN attack dataset. The dataset features contain

[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 paper.

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.

We also study the Survival Analysis datasets for automobile IDS which consist of Flooding, Fuzzy and Malfunction attacks in different car models: Sonata, Soul and Spark [2].

B. 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 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. Fig. 3 and Fig. 4 depict the attack injection scenarios on CAN messages.

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 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. 5(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 0×000 and $0 \times 7FF$ 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. 5(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.

TABLE 1. 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%)

TABLE 2. 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

Regarding the Survival Analysis datasets multiclass classification, we concatenate all the three attack classes for each car model. After concatenation, we observe from Table 2 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 2, 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.

IV. LSTM-BASED NETWORK INTRUSION DETECTION SYSTEM

For our investigation, we use python PyCharm IDE 2019.2.2 and Keras [21] 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 3 and 4 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 elements and we test the classifier with 20% of the elements. After training, the classifier can classify the attack and benign class elements. Fig. 10 schematizes the deep neural network model

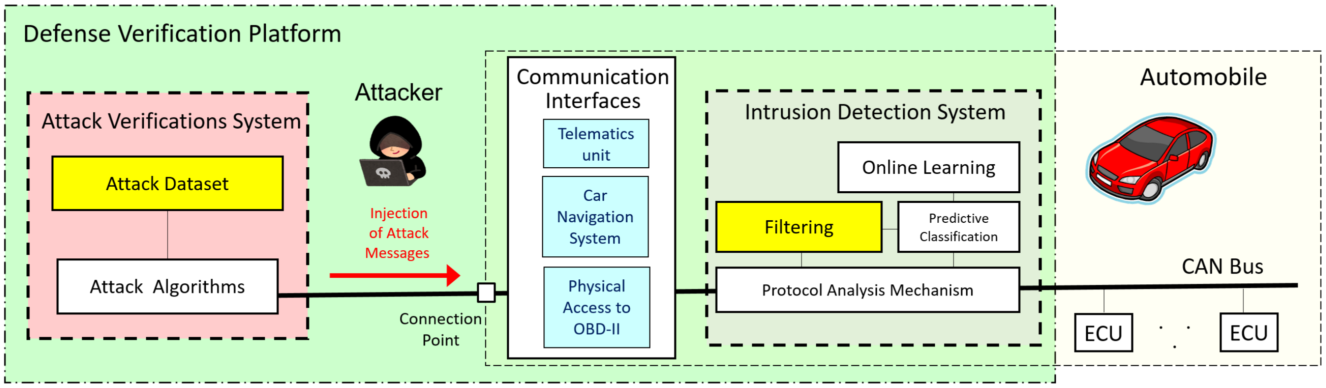

Defense Verification Platform
Attack Verifications System
Attack Dataset
Attack Algorithms
Attacker
Injection of Attack Messages
Connection Point
Communication Interfaces
Telematics unit
Car Navigation System
Physical Access to OBD-II
Intrusion Detection System
Online Learning
Filtering
Predictive Classification
Protocol Analysis Mechanism
Automobile
CAN Bus
ECU
ECU

FIGURE 6. CAN bus network system defense verification platform. Consist of two modules: Attack verification and intrusion detection system.

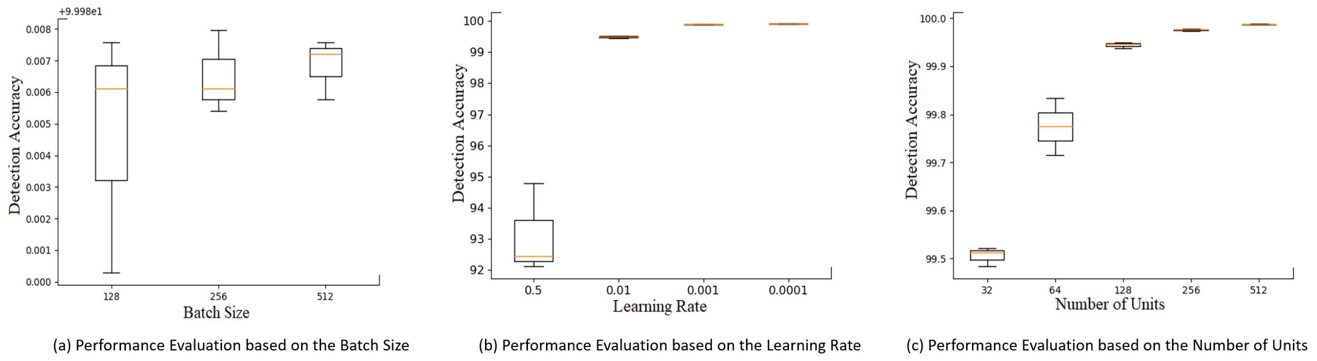


FIGURE 7. Performance Evaluation based on the Batch Size, Learning Rate and Number of LSTM Units.

TABLE 3. 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 4. 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

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 6 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.

A. 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 III, 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

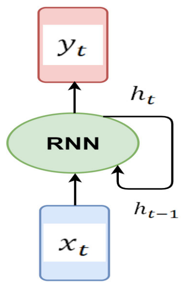


FIGURE 8. Basic RNN architecture.

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. 5; 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 1, 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.

B. 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 [18]. 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. 8), we can process a sequence of data $x_1, \dots x_n$ by applying RNN and it will produce a sequence of outputs $y_1, \dots y_i$.

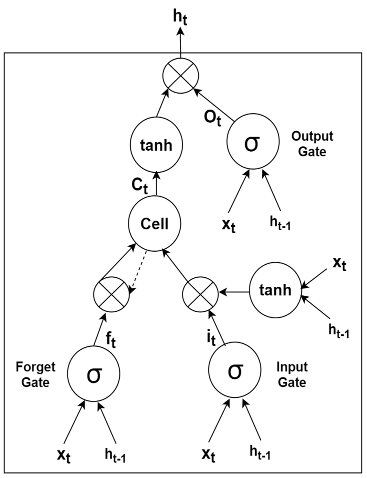


FIGURE 9. LSTM cell architecture.

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

h_t = New state
f_W = Function with parameter W
h_{t-1} = old state
x_t = input vector at time step t

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:

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b) \tag{2}$$
$$y_t = h_t \tag{3}$$

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 [19].

Three gates are available in LSTM and they are responsible for the cell state protection and control [20]. Figure 9 depicts the LSTM cell architecture consisting of the input vector x_t, hidden input vector h_{t-1} and output vector h_t.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{4}$$
$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{5}$$
$$\bar{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{6}$$

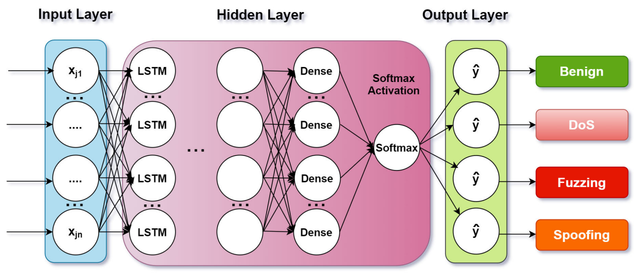


FIGURE 10. LSTM IDS architecture regarding the attack classification.

$$C_t = f_t * C_{t-1} + i_t * \bar{C}_t \tag{7}$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{8}$$
$$h_t = o_t * tanh(C_t) \tag{9}$$

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).

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

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

We give an input sequence $X = (X_1, X_2, X_3, \dots, X_i)$ and $X_j = (x_{j1}, x_{j2}, x_{j3}, \dots, x_{jn})$, $j \in [1, \dots, i]$, where x_jz, $z \in [1, \dots, n]$, is an element of the input variables (X), and we obtain an output sequence $Y = (y_1, y_2, y_3, \dots, y_i)$, where y_q, $q \in [1, \dots, i]$, 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. 7(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. 7(a) shows that 256-512 batch size provides better detection accuracy. Table 3 and Table 4 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.

In this experiment, we use the Keras [21] API. Keras is written in Python; hence, it is easy to use it along with Tensorflow. Table 3 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 parameters of the optimizer are used with their default values. In Fig. 7(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

TABLE 5. Binary classification results - NAIST CAN attack sataset.

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

units. In Section V, we discuss the details regarding how hyper-parameter values change the experiment settings and results.

V. 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 [22].

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 [23], [24].

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 3. Table 5 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.

A. 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

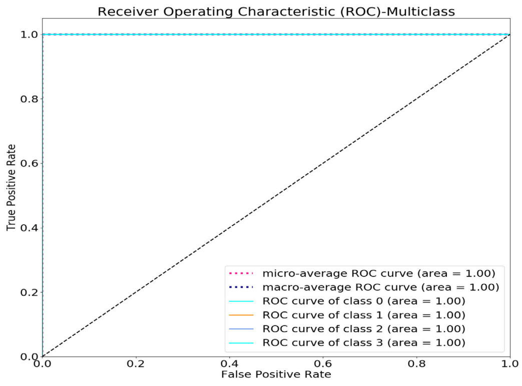


FIGURE 11. NAIST CAN attack dataset classification receiver operating characteristics (ROC).

values, we have to proceed with systematic experimentation for the particular predictive models [25].

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 4, and we compare the performance among them. In Table 6, 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 6 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.

TABLE 6. 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

Fig. 12 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

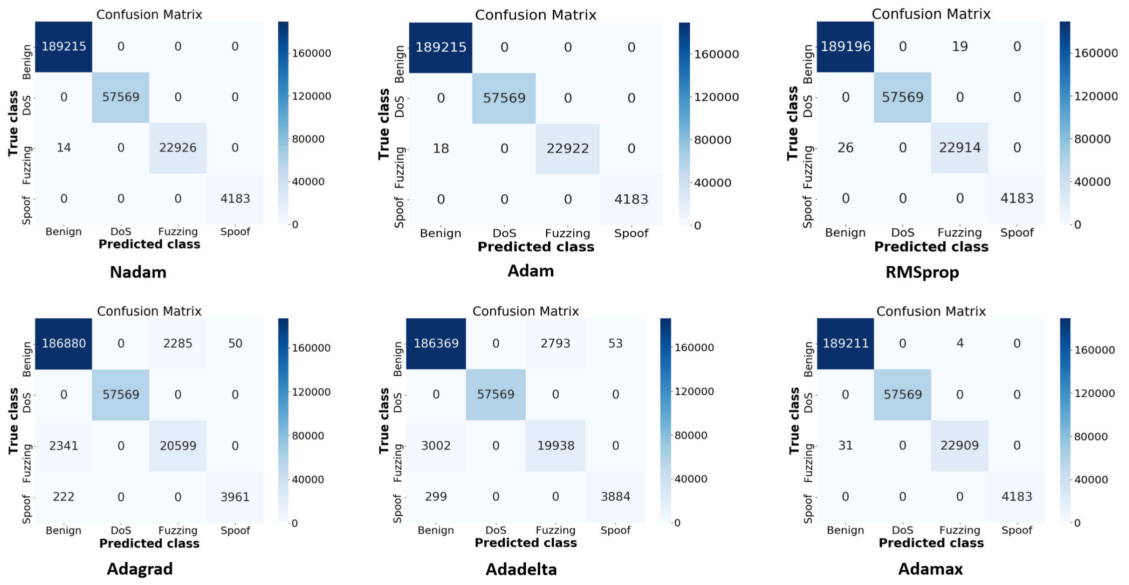


FIGURE 12. Gradient Descent optimizer confusion matrix - NAIST CAN attack dataset.

TABLE 7. 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

TABLE 8. 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

(14 instances) are available regarding the Fuzzing attack. Based on the CM and Table 7 (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 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 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. 11 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.

B. NADAM LEARNING RATE - LSTM CLASSIFICATION RESULTS

Gradient descent is one of the most popular and extensively used optimizer algorithms for optimizing neural networks [28]. 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 [26], [27]. We compare the performance by using a few learning rate values. Table 8 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.

C. 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

TABLE 9. 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 10. 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

data we would like to compute the gradient of the objective function [28].

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 9, *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 [29]–[31]. *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 12, we observe that *Nadam* and *Adam* classify most of the instances correctly with fewer false positives and false negative instances.

D. 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 settings in Table 4. In this experiment, *softmax* output activation function remains the same, we only change the input activation functions as *tanh*, *relu* and *sigmoid*. Table 10 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%.

E. 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 [32].

We also change the loss functions and compare the performance among them. Table 11 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 11. 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

TABLE 12. 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

F. RESULTS COMPARISON WITH THE SURVIVAL ANALYSIS METHOD/DATASET

We apply the proposed LSTM IDS to the Survival Analysis Dataset for automobile IDS [2]. The datasets contain three different car models with Flooding, Fuzzy, and Malfunction attacks. We use binary and multiclass classification according to Tables 3 and 4 parameter settings. From the experiment results, we observe that in Tables 12 and 13 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. 13). Our classifier can classify the Flooding and Malfunction attacks with a high detection rate of 1.00 (Table 13). 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.

VI. DISCUSSION

Researchers whose work revolves around deep learning face issues related to the availability of real-time public datasets.

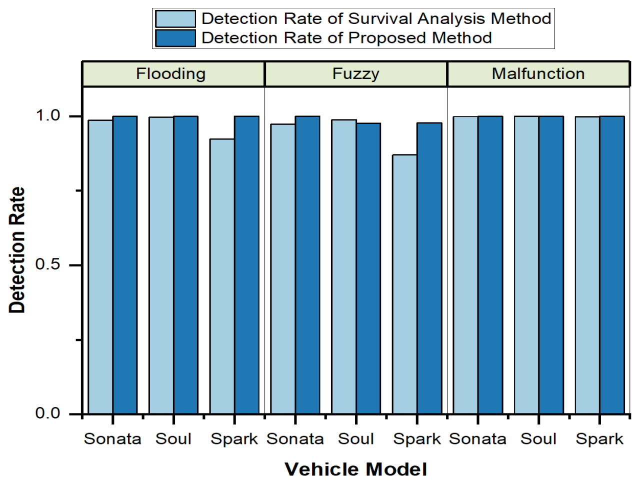


FIGURE 13. Detection rate - Comparison with the survival analysis dataset for automobile IDS.

TABLE 13. 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

We produce our CAN bus attacks 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 detection 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 paper.

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.

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.

VII. CONCLUSION

In this paper, 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.

REFERENCES

[1] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based intrusion detection system for in-vehicle network," in *Proc. 16th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2018, pp. 1–6.

[2] M. L. Han, B. I. Kwak, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," *Veh. Commun.*, vol. 14, pp. 52–63, Oct. 2018.

[3] X. Mo, P. Chen, J. Wang, and C. Wang, "Anomaly detection of vehicle CAN network based on message content," in *Proc. Int. Conf. Secur. Privacy New Comput. Environ.*, Cham, Switzerland: Springer, 2019, pp. 96–104.

[4] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy*, 2010, pp. 447–462.

[5] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "Long short-term memory-based intrusion detection system for in-vehicle controller area network bus," in *Proc. IEEE 44th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2020, pp. 10–17.

[6] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2923–2960, 4th Quart., 2018.

[7] S.-F. Lokman, A. T. Othman, and M.-H. Abu-Bakar, "Intrusion detection system for automotive controller area network (CAN) bus system: A review," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, p. 184, Dec. 2019.

[8] R. E. Haas, D. P. F. Moller, P. Bansal, R. Ghosh, and S. S. Bhat, "Intrusion detection in connected cars," in *Proc. IEEE Int. Conf. Electro Inf. Technol. (EIT)*, May 2017, pp. 516–519.

[9] P. Kleberger, T. Olovsson, and E. Jonsson, "Security aspects of the in-vehicle network in the connected car," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 528–533.

[10] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D. Gan, "Cloud-based cyber-physical intrusion detection for vehicles using deep learning," *IEEE Access*, vol. 6, pp. 3491–3508, 2018.

[11] M.-J. Kang and J.-W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," *PLoS ONE*, vol. 11, no. 6, Jun. 2016, Art. no. e0155781.

[12] M. Jaynes, R. Dantu, R. Varriale, and N. Evans, "Automating ECU identification for vehicle security," in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2016, pp. 632–635.

[13] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2016, pp. 63–68.

[14] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame," in *Proc. 15th Annu. Conf. Privacy, Secur. Trust (PST)*, Aug. 2017, pp. 5709–5757.

[15] Z. Khan, M. Chowdhury, M. Islam, C.-Y. Huang, and M. Rahman, "Long short-term memory neural networks for false information attack detection in software-defined in-vehicle network," 2019, *arXiv:1906.10203*. [Online]. Available: http://arxiv.org/abs/1906.10203

[16] S. Woo, H. Jin Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle CAN," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 993–1006, Apr. 2015.

[17] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *Proc. IEEE Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Oct. 2016, pp. 130–139.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 8, no. 9, pp. 1735–1780, 1997.

[19] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019.

[20] *Colah's Blog, Understanding LSTM Networks*. Accessed: Aug. 15, 2020. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[21] *Keras: The Python Deep Learning Library*. Accessed: Feb. 4, 2020. [Online]. Available: https://keras.io/

[22] Scikit-Learn Project. *Receiver Operating Characteristic (ROC)*. Accessed: Mar. 3, 2020. [Online]. Available: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

[23] R. Joshi. *Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures*. Accessed: Nov. 4, 2019. [Online]. Available: https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

[24] Koo Ping Shung. *Accuracy, Precision, Recall or F1*. Accessed: Nov. 1, 2019. [Online]. Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

[25] J. Brownlee. *How to Configure the Number of Layers and Nodes in a Neural Network*. Accessed: Mar. 2, 2020. [Online]. Available: https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/

[26] C. Darken, J. Chang, and J. Moody, "Learning rate schedules for faster stochastic gradient search," in *Proc. Neural Netw. Signal Process. II IEEE Workshop*, Aug. 1992, pp. 1–11.

[27] R. Gandhi. *A Look at Gradient Descent and RMSprop Optimizers*. Accessed: Nov. 5, 2019. [Online]. Available: https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b
[28] S. Ruder. *An Overview of Gradient Descent Optimization Algorithms*. Accessed: Nov. 2, 2019. [Online]. Available: https://ruder.io/optimizing-gradient-descent/
[29] Duchi, John, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. 7, pp. 1–39, 2011.
[30] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," 2012, *arXiv:1212.5701*. [Online]. Available: http://arxiv.org/abs/1212.5701
[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[32] J. Brownlee. *How to Choose Loss Functions When Training Deep Learning Neural Networks*. Accessed: Apr. 26, 2020. [Online]. Available: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

MD DELWAR HOSSAIN (Graduate Student Member, IEEE) received the M.Sc. degree in engineering in information systems security from the Bangladesh University of Professionals, in 2018. He is currently pursuing the Ph.D. degree with the Nara Institute of Science and Technology, Japan. He is also a Research Assistant with the Nara Institute of Science and Technology. His research interests include cybersecurity, deep learning, smart car security, smart grid security, industrial control systems security and smart city security.

HIROYUKI INOUE (Member, IEEE) received the Ph.D. degree in engineering from the Nara Institute of Science and Technology, in 2000. He has been an Associate Professor with the Graduate School of Information Science, Hiroshima City University, since 2010. His research interests include technologies for embedded security, especially automotive network security, and network protocol of the Internet.

HIDEYA OCHIAI (Member, IEEE) received the B.E., M.E., and Ph.D. degrees from The University of Tokyo, Japan, in 2006, 2008, and 2011, respectively. He is currently an Associate Professor with The University of Tokyo, where he is also the Head of LAN-Security Monitoring Project. His research interests include sensor networking, delay tolerant networking, and building automation systems, the IoT protocols, and cyber-security. He involves in the standardization of facility information access protocol in IEEE1888, ISO/IEC, and ASHRAE.

DOUDOU FALL received the M.E. degree in data transmission and information security from Cheikh Anta Diop University, Senegal, in 2009, and the M.E. and Ph.D. degrees in information science from the Nara Institute of Science and Technology (NAIST), Japan, in 2012 and 2015, respectively. He is currently an Assistant Professor with the Division of Information Science, NAIST. His research interests include cloud computing security, the IoT security, blockchain security, vulnerability, and security risk analysis.

YOUKI KADOBAYASHI (Member, IEEE) received the Ph.D. degree in computer science from Osaka University, Japan, in 1997. Since 2013, he has been working as the Rapporteur of ITU-T Q.4/17 for cybersecurity standardization. He is currently a Professor with the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. His research interests include cybersecurity, Web security, and distributed systems. He is a member of IEEE Communications society.