Deep Learning-Based Anomaly Detection for Connected Autonomous Vehicles Using Spatiotemporal Information

Pegah Mansourian, *Graduate Student Member, IEEE*, Ning Zhang, *Senior Member, IEEE*, Arunita Jaekel, and Marc Kneppers

***Abstract*— Although connected autonomous vehicles (CAVs) hold great potential to improve driving safety and experience significantly, cybersecurity remains a critical concern. As the de-facto standard for in-vehicle networks, the Controller Area Network (CAN) carries messages and commands vital to the operation of the vehicle. However, due to a lack of security mechanisms, intruders are able to conduct devastating attacks on drivers and passengers over CAN. In order to safeguard CAVs, an Intrusion Detection System (IDS) can be deployed to monitor CAN network activities and detect suspicious behavior resulting from an attack. This paper proposes a prediction-based IDS framework for detecting anomalies and attacks on a CAN bus using temporal correlation of message contents. Two candidates are introduced as the prediction module. The first network is an LSTM that predicts time series data separately for each CAN ID, and the second is a ConvLSTM that predicts messages using correlated data of several CAN IDs. An attack is classified according to prediction errors by a Gaussian Naïve Bayes classifier. The proposed IDS is evaluated against other state-of-the-art one-class classifiers, including OCSVM, Isolation Forest, and Autoencoder, and three existing works, including ReducedInception-ResNet, NeuroCAN, and CANLite, using a real-world dataset, the Car Hacking Dataset. A comparison between the two suggested architectures and their use cases is given. Compared to baseline methods and related studies, the proposed method is shown to be more accurate and can achieve F-scores and detection accuracy of almost 100%.**

***Index Terms*— In-vehicle security, CAN, anomaly detection, IDS, LSTM, ConvLSTM, spatiotemporal correlation.**

I. Introduction

CONNECTED autonomous vehicles (CAVs) equipped with a variety of sensors and onboard computing, will contribute to significant improvements in road safety and efficiency, playing a vital role in intelligent transportation systems (ITS) [1], [2]. In a typical CAV, there are over 100 electronic control units (ECUs), allowing a certain level of automation and enabling vital information to be gathered for decision-making. ECUs can handle various tasks. These include simple operations such as opening a window to more critical and complex ones that need communication between several ECUs, such as lane departure and adaptive cruise control. An in-vehicle communication network enables the ECUs to exchange messages with each other and with the gateway to the outside world to ensure the vehicle operates properly. A variety of protocols have been designed for the in-vehicle network, namely Local Interconnect Network (LIN), Media-Oriented System Transport (MOST), FlexRay, and Controller Area Network (CAN) [3]. A high-bandwidth protocol such as MOST is mainly used for infotainment systems, while a low-bandwidth protocol like LIN is used for equipment not critical to safety. The simplicity and reliability of CAN, however, make it the dominant protocol used in vehicles.

While CAV introduces a range of comfort and safety features, all of the added components, including ECUs, distributed communications, and external wireless access to the intra-vehicle network, result in new attack surfaces and incur security issues [4], [5]. Through the use of external wireless communication, such as Wi-Fi and Bluetooth, or through physically tapping into the CAN bus via the OBD-II Port, an attacker can gain access to the vehicle's network and cause serious damage to the driver, passengers, and vehicle. A CAN message is not intrinsically encrypted, authenticated, or provided with other security measures. With the CAN bus being a broadcast media, lacking authentication (sender ID) and encryption, and weak access control, the CAN protocol is vulnerable to many attacks, including DoS, masquerades, suspensions, replays, fabrications, and remote access attacks. A CARSHARK component was used to sniff packets on a CAN bus in an experiment by Koscher et al. They were then able to identify the corresponding messages of each ECU using targeted probing, fuzzy, and reverse engineering techniques. The hackers managed to override messages sent to door locks, headlights, and wipers in the Body Control Module (BCM) and displayed their fake speedometer readings on the Driver Information Center (DIC) [6].

In-vehicle networks, including the CAN bus, can be safeguarded using three main mechanisms: preventative measures like authentication and encryption, detection measures like intrusion detection systems (IDS), and mitigation and recovery

Manuscript received 29 December 2022; revised 1 May 2023; accepted 25 May 2023. Date of publication 27 June 2023; date of current version 29 November 2023. This work was supported by Mitacs through the Mitacs Accelerate Program. The Associate Editor for this article was Z. Lv. *(Corresponding author: Ning Zhang.)*

Pegah Mansourian and Ning Zhang are with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B 3P4, Canada (e-mail: ning.zhang@uwindsor.ca).

Arunita Jaekel is with the Department of Computer Science, University of Windsor, Windsor, ON N9B 3P4, Canada.

Marc Kneppers is with TELUS, Vancouver, BC V6B 3K9, Canada.

Digital Object Identifier 10.1109/TITS.2023.3286611

measures like node isolation. Authentication mechanisms verify both the message content and identity of the sender of the CAN messages, to prevent unauthorized or compromised ECUs from sending malicious messages. Automotive Open System Architecture (AUTOSAR) [7] provides a set of standards for the in-vehicle authentication mechanism that inspires many lightweight authentication methods for the CAN bus [8], [9], [10]. Encryption methods conceal the content of the message, making them more resilient to malicious modifications. Both authentication and encryption mainly rely on symmetric and asymmetric cryptography, which increases the communication and computation overhead. Also, appending a Message Authentication Code (MAC) to the limited-length CAN message is another limitation of these attack prevention methods.

Due to the absence of intrinsic security mechanisms in the CAN protocol, and the inefficiency introduced by cryptography-based methods, the next defense layer is the deployment of an IDS in accordance with the defense-in-depth strategy. Furthermore, an intruder may bypass attack preventative mechanisms and gain access to the ECUs. An attack detection mechanism is therefore needed to detect these intrusions and provide CAN security. The purpose of an IDS is to monitor events and messages in a network in order to detect malicious activities and violations. A mitigation and recovery mechanism can be applied when an attack is detected by an IDS. A mitigation procedure may involve isolating the compromised ECU or CAN bus from the rest of the in-vehicle network and informing the driver and manufacturer. The logs of traffic can be used for in-depth analysis and forensics in order to address design flaws that may compromise security.

IDSs can be classified as either signature-based, anomaly-based, or hybrid depending on how it recognizes threats. For signature-based IDS, every network traffic is verified against a database containing known attack signatures to find out if there is a match or if an intrusion has occurred. It is not possible to detect unknown attacks or zero-day attacks using signature-based IDS. In contrast, anomaly-based IDS achieves this objective by detecting deviations from the expected network profile as intrusions. Due to the ability to analyze rarely available attack data in an independent manner without requiring any human knowledge, it is becoming more popular these days. Some anomaly-detecting IDSs use the statistical [11], [12] or information theory [13] features of CAN messages to identify anomalies, while others offer machine learning methods like SVM [14], [15], [16], HMM [17], [18], SOM [19], GBDT [20], and NN [21], [22], [23], [24], [25], [26], [27], [28], [29], [30].

In this paper, an anomaly-driven IDS framework for the CAN bus network is presented which uses machine learning methods. The attacks on the CAN bus result in changing the time series data of message contents to appear as grouped (conditional) anomalies. They can only be identified when they are examined as part of a batch of results. To address this issue, a time-series prediction network is proposed to determine the normal model of the CAN network based on the sequence of messages over time. CAN messages from the past are used as input for predicting the current expected message. The conditional anomalies in CAN messages can be reduced to point anomalies by considering prediction errors, and then classified by using conventional machine learning methods. In the absence of an attack, the predicted value matches the given value. Prediction errors signal suspicious activity when it comes to an attack. It has been found that prediction error distributions for normal and attack classes follow Normal or Gaussian distributions with different means and standard deviations. Gaussian Naive Bayes (GNB) is a classifier that focuses on the difference between Gaussian distributions across classes. The prediction module has therefore been combined with a GNB classifier to find point anomalies. For the time-series prediction module, an LSTM network is designed first, which operates on the sequential CAN messages of each ECU, and makes a prediction of the current message of the ECU, independent of the others. Additionally, a correlation appears to exist between the time series of the ECUs since they collaborate to perform in-vehicle tasks. For example, when the brake is pressed, the brake ECU sends an activation message, the acceleration ECU sends a negative value, the speed ECU sends a decreasing value, and the brake light ECU sends a signal to turn on. ConvLSTM is another temporal forecasting model suggested in the proposed framework to use the spatial correlation among messages with different CAN IDs. While the ConvLSTM prediction model saves memory and processing resources for detecting anomalies, its performance depends more than the LSTM model on the amount of correlation among messages.

The remainder of the paper is organized as follows. A review of previous studies on CAN bus anomaly detection-based IDS, categorized by their methodologies, is provided in Section II. Then, each building block of the proposed IDS framework is described in detail in Section III. In Section IV, two proposed prediction modules are discussed and compared. Experimental results are presented in Section V, as well as comparisons with state-of-the-art baselines and existing studies using the Car Hacking Dataset. Finally, conclusions are provided in Section VI.

II. Related Works

As the dominant in-vehicle communication protocol, CAN was designed to be lightweight, robust, and real-time. Therefore, no security mechanisms are implemented into it. CAN's simplified structure and lack of security insight have made it vulnerable to attack and interruption, either physically or remotely. In response, the researchers investigated in-vehicle security and designed IDS solutions for vehicles.

Based on the methodologies used in attack detection, the studies can be categorized into statistical, information theory, and machine learning-based IDS. The statistical methods mainly focus on the frequency and timing feature of messages and using the statistics summary of them, like count, average, and standard deviation, extract a profile of normal observations on the CAN bus [11], [12]. The change in the network's entropy is utilized in the information theory methods as the differentiation strategy of normal and attack situations [13].

In spite of the fact that statistical and entropy-based approaches are very lightweight and resource-efficient, they may not be able to detect unseen (zero-day) attacks, especially when they occur within the content of messages. It has become increasingly popular to apply machine learning to solve problems arising in unknown and non-linear environments to overcome this challenge. In this way, the patterns and features of the system can be defined without any prior expert knowledge. Recent studies on CAN bus anomaly detection have extensively used it. Machine learning-based methods include some that use conventional machine learning methods. The most popular conventional algorithms are SVM [14], [15], [16], HMM [17], [18], SOM [19], and GBDT [20].

Studies on the detection of CAN bus anomalies have shown positive results using neural networks. A DNN structure was proposed by Kang et al. to detect attacks on the CAN bus. Restricted Boltzmann Machines are used to pre-train the weights of DNN models to prevent the vanishing gradient problem. With seven hidden layers of deep networks, they achieved a 99.9% detection rate with a simulated dataset generated by OCTANE [21].

An autoencoder with a denoising structure based on deep neural networks was proposed by Lin et al. Anomalies cannot be reconstructed perfectly due to the normal data used to train the model. Therefore, it raises an anomaly flag if there is an error in reassembling the anomalies. A further novelty is that an evolutionary optimization algorithm has been used to determine the optimal amount and size of hidden layers in the network. Tests were conducted with three datasets - two generated and one publicly available OTIDS dataset. In terms of precision, recall, and F-score, it was compared to ANN, K-means, and Decision Tree and achieved an F-score of 0.98, which is better than the baselines [22].

Considering that CAN bus messages are correlated in time, Taylor et al. incorporated LSTM into their network. Each ECU's next expected DATA payload is predicted separately, and a bit log loss error is aggregated over the whole field to detect anomalies. They achieved approximately 0.99 AUC by performing this experiment using data collected from a 2012 Subaru Impreza that had five types of anomalies: interleave, drop, discontinuity, unusual, and reverse [24].

Zhu et al. also use LSTM for anomaly detection. In order to reduce response time and computation power, they distributed their LSTM training to edge devices [25]. An LSTM-based autoencoder based on CANnolo was proposed by Longari et al. For the detection of anomalies in independent ECUs and validated on their own dataset of CAN messages [26].

TENET redesigns the CNN and converts it into a temporal CNN to include the sequential analysis of CAN messages in its prediction [31]. TENET is composed of TCNA (temporal convolutional neural attention) residual blocks stacked together. The difference between predicted and actual signals is computed with a divergence score and classified by a decision tree. The method is tested on the SynCAN dataset containing Plateau, Suppress, Continuous, and Playback attacks.

Anomalies have been found in the above-mentioned studies when looking at time series of messages with different CAN IDs separately, but other studies have focused on the correlation of data from ECUs for the detection of anomalies. The methods used in these studies vary from adapting time series forecasting solutions to more dedicated methods, namely CNN-LSTM and ConvLSTM.

One of the adaption methods is the CANet proposed by Hanselmann et al. In this IDS, the LSTM model is adjusted by concatenating the LSTM predictions of separate CAN IDs and reconstructing the input by feeding the joint vector into an autoencoder. This way they could take the impact of messages with other CAN IDs on the current message into account. They used a fixed threshold on the reconstruction error of the autoencoder part as the anomaly detector. As a result of the evaluation, it was found that reconstructing CAN messages based on all CAN IDs yielded better results than analyzing them separately [27]. In this study, however, the neural network structure design is based on decoded CAN messages from the SynCAN dataset, which is derived from the communication matrix (or CAN matrix).

In [29], Balaji and Ghaderi proposed another LSTM adaption approach, called NeuroCAN, to detect anomalies in CAN messages by incorporating inter-ECU dependencies. They achieved this by introducing an embedding layer to predict the upcoming message, which was then followed by an LSTM and an output layer. The embedding layer was created through a linear transformation of input data from each CAN ID, passed through a sigmoid function, and then accumulated over all IDs. However, training the network separately for each CAN ID led to high memory and computational costs.

To address this issue, the authors developed an improved version of NeuroCAN, called CANLite. Rather than training the network separately for each CAN ID, CANLite uses a single multi-task model. This model extracts spatial and temporal dependencies of CAN IDs and predicts the next payload of a requested CAN ID based on a prediction parameter. This results in a more lightweight contextual anomaly detection system, which is better suited for CAN buses with limited resources and detection time. By using CANLite, the authors were able to achieve accurate detection of anomalies, while minimizing memory and computational costs [30]. One advantage of this work's approach is that it is relatively straightforward to implement since it only requires setting a threshold based on the MSE metric that yields the highest F-score. However, this method may not be as effective at detecting more complex and subtle anomalies.

Song et al. suggested a modified Inception-ResNet that employs CNN units and was trained on the CAN ID sequence observed on the bus. This was done in order to differentiate between normal and attack sequences [28]. Although they were successful in achieving high performance, their approach did not take into account the content of messages and was trained on attack data. Therefore, it may not be effective for detecting new, previously unseen attacks.

The purpose of CANintelliIDS, developed by Javed et al., is to improve detection performance by correlating data from multiple ECUs. A continuous stream of CAN messages has been processed by a neural network consisting of two layers of CNN followed by a GRU layer. Compared to conventional ML and pure CNN, the proposed method performed better on the OTIDS dataset, with a 93% F-score [23]. While the presented work in this study has a good performance, its goal is to categorize the CAN traffic into four pre-defined classes: attack-free, DoS attack, Fuzzy attack, and malfunction attack messages. However, it is important to note that this method is limited in that it cannot identify unknown or zero-day attacks.

Sun et al. proposed an anomaly detection system, called CLAM, for CAN bus, using a combination of CNN, LSTM, and attention mechanism. The structure is a multi-branch neural network. The input entries from all CAN IDs are put together in a matrix to make a two-dimensional time series and fed into a Convolution layer to extract its spatial features. Then the extracted feature maps are fed into a bidirectional LSTM. The LSTM outputs are concatenated with an attention mechanism and passed through a dense layer for prediction. The RMSE is used as the anomaly score and a fixed threshold is defined to announce anomalies [32]. Similar to other methods that rely on setting thresholds, this approach may not be as efficient in detecting complex and subtle anomalies.

Xiao et al. introduced an IDS for the in-vehicle CAN network which uses a ConvLSTM unit in its structure. ConvLSTM is an extension of the LSTM unit, designed for extracting the local dependency of the inputs and the temporal one. A threshold approach was used to classify anomalies based on Pearson correlation coefficient between predicted and actual messages. Observing that the correlation coefficient gradually declined over time, they proposed an adaptive threshold based on the average of 50 consecutive coefficients. The approach was tested on the OTIDS dataset and performed better than the decision tree and the SVM [33]. The adaptive and self-evolving threshold-setting method is fully data-driven and does not require human knowledge, but it still suffers from the deficiencies of threshold-based detection methods.

When compared to other methods, neural networks demonstrate better results in identifying attacks on in-vehicle networks. Specifically, time-series analysis techniques like LSTM are particularly suitable in this scenario. Nevertheless, these methods have limitations. To begin with, some of them are supervised methods that require labeled attack data to classify the attacks. Therefore, a substantial amount of attack data is required during the training process to establish a dependable model, which is inefficient because attack data is not readily available in the real world. Furthermore, these methods cannot identify previously unseen attacks. Additionally, prediction-based anomaly detection techniques in the literature define an anomaly score and establish a threshold to detect anomalies. However, this threshold-based detection approach is not always able to identify subtle deviations accurately, and human intervention may be necessary.

We have suggested a method for addressing these obstacles, which involves a time-series analysis of CAN messages to identify anomalies. Our contributions are as follows:

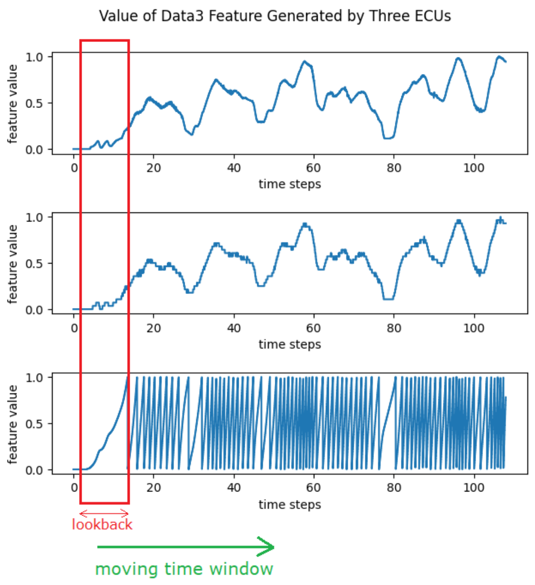


Fig. 1. The creation of sequential input from correlated CAN messages. The Data3 attribute is displayed, which is the fourth byte of the CAN payload, for three distinct CAN IDs.

- Our approach to anomaly detection utilizes the normal profile of messages on the CAN bus for making predictions of the upcoming message based on the seen historical messages.
- We have designed two prediction models, one based on LSTM and the other on ConvLSTM. These models take advantage of the temporal and spatiotemporal correlations of CAN messages. The LSTM module predicts the next messages produced by each ECU individually and without considering the others. On the other hand, the ConvLSTM module can utilize the spatiotemporal correlation of messages with various CAN IDs to consider the interdependence of different ECUs.
- Our proposed framework utilizes a Gaussian Naive Bayes classifier to classify MAE as the prediction error metric, which can be more effective at detecting subtle anomalies that may not be detected by the threshold setting method.
- Our method directly processes the bit representation of the payload. This makes it easier to generalize across different manufacturers and vehicles.
- Due to its anomaly detection approach, the proposed IDS is capable of detecting both known (such as DoS, Fuzzy, and Spoofing) and unknown attacks.

III. Proposed CAN Bus Anomaly Detection Framework

A timer, an event, or both may trigger a CAN message. ECU measurements are periodically broadcast via time-triggered CAN messages. Also, some events can trigger an ECU to generate a CAN message. Fig. 1 illustrates the time series data of the fourth byte of the payload, referred to as "Data3", for three distinct CAN IDs. As can be seen in this figure, the contents of messages are correlated over time and among ECUs. Variations observed in the data of the first ECU led to comparable patterns in the second ECU. Additionally, a decrease in the data of the first ECU caused an increase in the gap between the peaks of the third ECU data. In fact, at every time, the payload of transmitted messages is influenced by the previous messages on the CAN bus. Based on this observation, we propose using a time series prediction module in our anomaly detection framework.

The complete structure of the proposed framework is depicted in Fig. 2. It consists of three modules: prediction network, prediction error calculator, and Gaussian Naïve Bayes (GNB) classifier. The input to our IDS is sequential data, and the output is the class label (attack or normal) with the highest probability of samples belonging to it.

The proposed IDS takes the messages from all CAN IDs as input for their detection. So they have to be deployed in a central node that can handle all messages, such as a CAN gateway or the onboard unit. The gateway is a more resourceful node, which has access to all messages on the bus and is responsible for translating messages between CAN and the outside network. Fig. 3 shows the deployment scenario of our time series prediction-based IDS design. We will discuss the building blocks of the proposed framework in detail.

A. Time Series Prediction Network

In general, time series are sequences of data points taken over a period of time at evenly spaced intervals. The analysis of time series can reveal how the historical observations of the phenomenon and other external factors affect future potential outcomes. In this process, a model is developed based on the recorded historical data to have an understanding of "why" a data point appears in a certain place of the sequence. Based on this model, an extrapolation is done to predict future possible observations.

Non-recurrent neural networks, such as MLP and CNN do not have memory in their structure and predict the output only based on the given input at the current time. So, they are not suitable for applications where data has sequential properties over time. The CAN message payload generated by an ECU can be viewed as a sequence and its value changes based on previous observations. A class of neural networks, known as Recurrent Neural Networks, RNN for short, provided the required ability, by implementing a memory cell in its structure holding the hidden state of previously observed data, to make predictions based on it.

Input data consists of time series information of CAN messages, which is then analyzed by the RNN network to learn how the data varies over time. During the data preparation stage, features are selected from CAN messages to identify their discriminating characteristics. These features include the Data Length Control (DLC) and eight bytes of payload data. The frames with a DLC that is less than 8 are padded with 0 to make it 8 bytes long. The min-max technique is used to normalize the dataset feature-wise due to the different scales of each feature. This increases the numerical stability and speed of the learning process. The min-max scaler transforms the data using the following formula:

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}. \quad (1)$$

After normalization, all the features have similar value ranges between zero and one. As represented in Fig. 1, a moving time window is defined to arrange the historical observations of x_t and generate a sequential input. The sequential input, $X = [x_{t-lookback}, \ldots, x_{t-2}, x_{t-1}]$, with lookback being the size of it, is then fed into the RNN-based network for processing.

The prediction module in our proposed framework is an RNN-based network. An RNN unit is distinguished by the memory cell in its gated structure. The output of an RNN unit is dependent on the preceding elements in a sequence. Weights learned during training in each gate are shared over time and determine how much information should be retained, as well as how much should be forgotten. The backpropagation procedure in RNN, called BPTT, is also slightly different as the calculated gradients propagate back through time. Training of the network is done using data in a normal scenario, containing no attack data. This procedure allows the extraction of a model of typical CAN messages. After the model is trained, it is deployed in a test environment, where some messages have been maliciously modified by an attacker, and the model is used to predict the probable message based on the normal profile and previous observations.

B. Prediction Error Calculator

Once the model is trained on the normal behavior of CAN IDs, it can be used to distinguish between normal and attack data in terms of prediction errors. Based on a sequence of historical data $[X_{t-lookback}, \ldots, X_{t-2}, X_{t-1}]$ taken into account by the time series prediction network as input, the expected status of each CAN message is predicted at time t, indicated by $\hat{X}_t$. To measure the dissimilarity of predicted and real observed values, Mean Absolute Error (MAE) is used in the prediction error calculator, which can be defined as:

$$MAE = \frac{\sum_{t=1}^{n} |X_t - \hat{X}_t|}{n}. \quad (2)$$

Measured prediction error indicates how closely the prediction network's output matches the current CAN message value. Since the forecast is done according to the normal profile of the network, it tends to be nearly zero for unmodified messages and relatively high for manipulated ones.

C. GNB Classifier

In a typical neural network-based anomaly detection, it is assumed that prediction errors for the normal data points follow a Gaussian distribution with an average of μ and standard deviation of σ, and any entry falling outside the confidence intervals ($\mu \pm n\sigma$, $n \in \mathbb{N}$), is regarded as anomalies. Researchers generally find it to be a powerful and popular approach, but determining a threshold or confidence interval to maximize the performance metrics for all attack types is a challenging and sometimes contradictory process. Also, in many cases, it requires an aggregation of errors among the dimension of the predicted data, which may result in losing the separability of classes.

Our study revealed that in addition to the prediction error of normal sequences, the prediction error of anomalous data

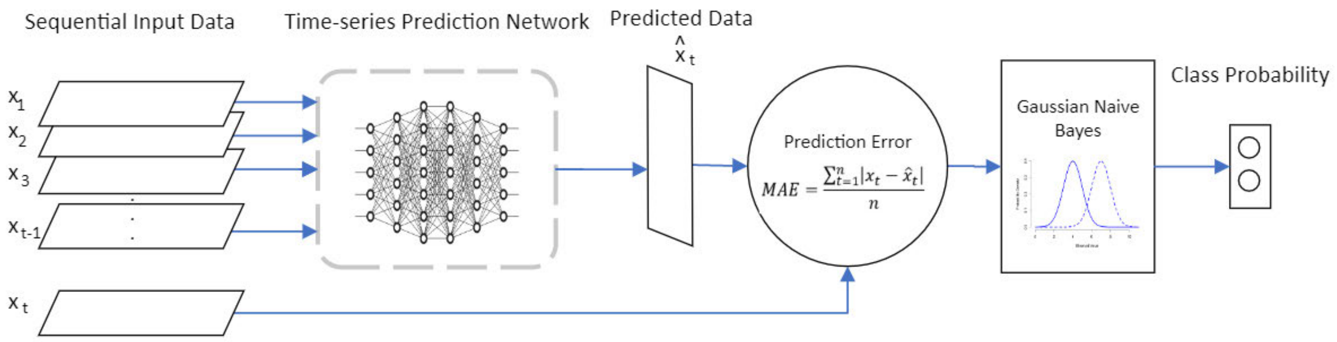


Fig. 2. The overall structure of the proposed CAN bus anomaly detection-based IDS framework.

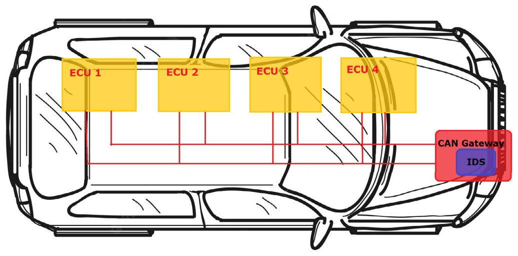


Fig. 3. IDS deployed in the gateway.

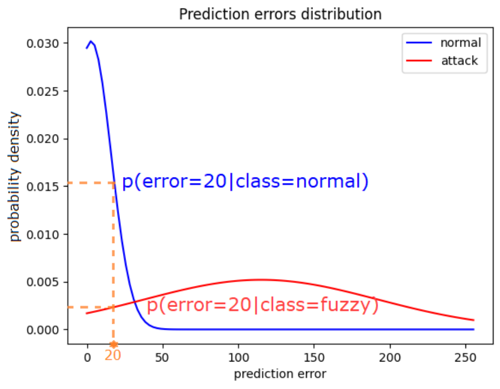


Fig. 4. The prediction error distribution of normal and attack data. The probability of a prediction value of 20 belonging to the normal class is higher than belonging to the fuzzy attack class.

points also follows a Normal bell-shaped distribution, but with different averages and standard deviations. The differentiation between the distribution of two classes could help to classify them more accurately. Since the primary purpose of adopting neural networks into anomaly detection is to automate the process and eliminate the necessity of human expert domain knowledge, in our study we proposed to integrate a Gaussian Naïve Bayes (GNB) classifier into the designed IDS structure.

Naïve Bayes is a supervised classifier algorithm designed based on the Bayes theorem and a Naïve assumption of conditional independence between features of the input data. Given class label y and the dependent feature vector x_1 through x_n, the Bayes' theorem is stated as:

$$p(y|x_1, \ldots, x_n) = \frac{p(x_1, \ldots x_n|y)p(y)}{p(x_1, \ldots, x_n)}, \tag{3}$$

where $p(y|x_1, .., x_n)$ is the conditional probability of class label y by considering data features $x_1, \ldots x_n$, and $p(y)$ and $p(x_1, \ldots x_n)$ as prior probability of class y and feature vector $x_1, \ldots x_n$. Assuming that features are conditionally independent, we will have:

$$p(y|x_1, \ldots, x_n) = \frac{\Pi_{i=1}^{n} p(x_i|y)p(y)}{p(x_1, \ldots, x_n)}. \tag{4}$$

The $p(x_1, \ldots, x_n)$ is constant, so it is discarded in the comparison. After calculating the posterior probability of all possible classes based on the prior possibilities, derived from the given dataset, the class label can be assigned as the one that maximizes the conditional probability of $p(y|x_1, .., x_n)$:

$$\hat{y} = \arg\max_y \Pi_{i=1}^{n} p(x_i|y)p(y), \tag{5}$$

where $\hat{y}$ is the predicted class for the data sample $x = [x_1, \ldots, x_n]$.

Gaussian Naïve Bayes is an extension of the Naïve Bayes classifier, assuming the real-valued input data features have Gaussian distribution with different parameters in different classes. Defining each class y by the average and standard deviation parameters, denoted by μ_y and σ_y, respectively, the conditional prior probability of $p(x_i|y)$ can be calculated as:

$$p(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp \frac{-(x_i - \mu_y)^2}{2\sigma_y^2}. \tag{6}$$

In case an attacker injects, drops, or manipulates the content of a CAN message, the affected message does not adhere to the normal sequence of data learned by the prediction module. Hence, the model fails to predict it perfectly, and the resulting prediction error is unconnected to the normal case. Consequently, as is apparent in Fig. 4, the distribution of the attack and normal data prediction errors, follow different Gaussian distributions. An example error value of 20 is highlighted in this figure. The GNB module compares the probability of the error belonging to each class and reports the one with a higher probability as the predicted class for it.

The GNB classifier module makes the final decision about the input message x_t, at time step t. It calculates the probability of the data belonging to classes of normal or attack and selects the class with a higher probability as the reported class. If the observed CAN message is identified as malicious, our IDS will raise the alarm and notify the driver and the manufacturing company.

IV. Prediction Module Neural Network Design

The overall structure of the proposed CAN bus anomaly detection-based IDS was described in the last sections. Based on the proposed framework, an RNN-based prediction network should be employed to extract the temporal correlation of CAN messages and build a model. We have suggested two architectures for the time-series prediction module: LSTM-based and ConvLSTM-based networks. The principles and structure of each are described in the following.

A. LSTM-Based Network

LSTM, an enhanced version of RNN, was introduced by Hochreiter and Schmidhuber [34] in 1997. LSTM solves the vanishing gradient problem of RNN by switching RNN's learning strategy from learning what information to remember to learning what information to forget. To do so, three learnable gates in the LSTM structure control the flow of information into the hidden state. The forget gate determines how much of the information about the past should be discarded. The amount of data flow from current input x_t and cell state c_t is controlled by the input gate based on their corresponding weights. Finally, the output gate allows the unit's current state to be outputted. Assuming that i_t, f_t, o_t, c_t, and h_t denote the input gate, forget gate, output gate, cell state, and hidden state at timestep t, respectively, the mathematical working structure of LSTM can be calculated as:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \odot C_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \odot C_{t-1} + b_f) \\ C_t &= f_t \odot C_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \odot C_t + b_o) \\ h_t &= o_t \odot \tanh(C_t), \end{aligned} \tag{7}$$

where $W_h = \begin{pmatrix} W_{hi} \\ W_{hf} \\ W_{hc} \\ W_{ho} \end{pmatrix} \in \mathbb{R}^{\text{dh}\times 4\text{dh}}$ is hidden-to-hidden weight matrix, $W_x = \begin{pmatrix} W_{xi} \\ W_{xf} \\ W_{xc} \\ W_{xo} \end{pmatrix} \in \mathbb{R}^{\text{dx}\times 4\text{dh}}$ is input-to-hidden weight matrix, $b = \begin{pmatrix} b_i \\ b_f \\ b_c \\ b_o \end{pmatrix} \in \mathbb{R}^{4\text{dh}}$ is the bias, and $C_0, h_0 \in \mathbb{R}^{4\text{dh}}$ are the initial value of cell state and hidden state, respectively. Also, $\sigma(.)$ denotes the sigmoid activation function and $\odot$ is the Hadamard product operator.

The LSTM prediction network, as depicted in Fig. 5 consists of one layer of LSTM with 32 units, followed by a fully connected layer with 9 neurons, each representing one output predicted feature. The activation function in the LSTM layer is tanh to account for the non-linearity of the temporal and long-term dependencies of CAN messages. To avoid the overfitting problem, we kept the network structure as simple as possible and employed an early stopping mechanism while training.

LSTM units are designed to learn and forecast one-dimensional time series data. Hence, in our case, one LSTM network is trained for each CAN ID separately. Separate error calculator and GNB classifier modules also process the prediction results of each of the LSTM-based detection systems. The CAN messages are first categorized based on their CAN ID and then redirected to the corresponding IDS. The IDSs operate independently of the others and flag anomalies once they are detected.

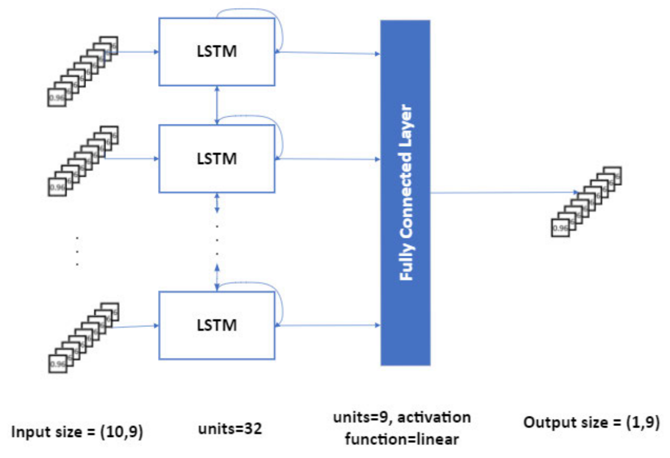


Fig. 5. LSTM-based prediction network architecture.

B. ConvLSTM-Based Network

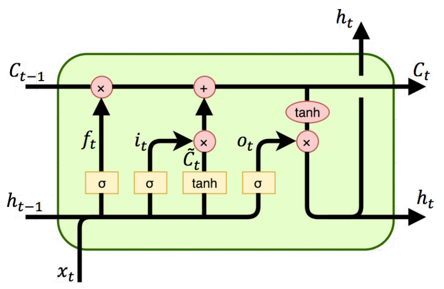
Besides the LSTM-based prediction model trained on the normal data of individual CAN IDs, an alternative architecture is proposed for detecting anomalies. To predict the upcoming message, this new neural network aggregates all the separate LSTM models into one central model and leverages the temporal correlation between all CAN IDs.

To incorporate the spatial correlation into the LSTM, Shi et al. proposed a modified version of LSTM that uses the convolution operation in its structure [35]. In this new structure, called ConvLSTM, the multiplication operation is replaced by the convolution operation. Compared to LSTM, ConvLSTM is better at extracting spatiotemporal correlations from input data with fewer model parameters. It determines the future state of a cell, based on its previous states and its local neighbors' inputs. The key equations of ConvLSTM can be formulated as follows:

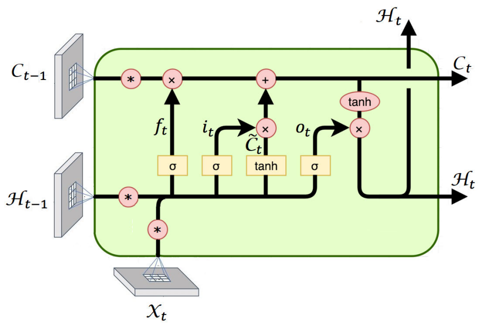
$$\begin{aligned} i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \odot \mathcal{C}_{t-1} + b_i) \\ f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \odot \mathcal{C}_{t-1} + b_f) \\ \mathcal{C}_t &= f_t \odot \mathcal{C}_{t-1} + i_t \odot \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \\ o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \odot \mathcal{C}_t + b_o) \\ \mathcal{H}_t &= o_t \odot \tanh(\mathcal{C}_t). \end{aligned} \tag{8}$$

Similar to LSTM equations given in Eq. 7, W_h and W_x are hidden-to-hidden and input-to-hidden weight matrices, respectively, and b is the bias matrix. Also, $\sigma(.)$ denotes the sigmoid activation function, and $\odot$ is the Hadamard product operator. What makes ConvLSTM different than LSTM is $\mathcal{X}$, $\mathcal{H}$, and $\mathcal{C}$, which are the spatiotemporal matrices of input, hidden state, and cell memory, respectively, and the convolution operator between weights and spatiotemporal matrices denoted by $*$. A comparison of the internal structure between LSTM and ConvLSTM units is given in Fig. 4.

There are two implementation variants for ConvLSTM - 1D and 2D-ConvLSTM. For the latter, the input is a 3D matrix, in which the first two dimensions are space and the third is features, also called channels, while the former has only



(a) LSTM unit



(b) ConvLSTM unit

Fig. 6. Internal structure of LSTM and ConvLSTM units.

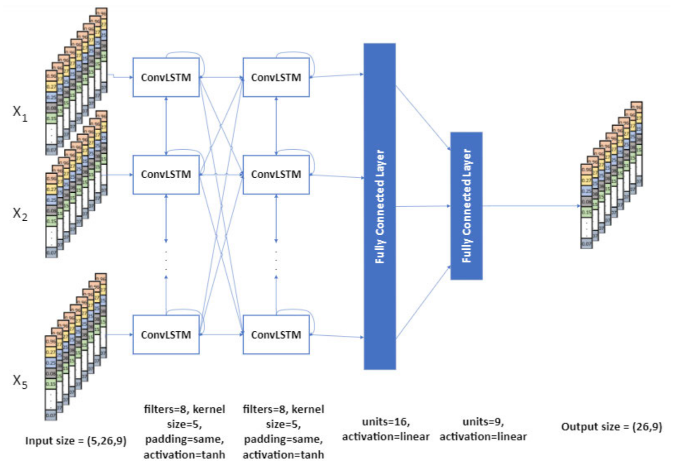

X1
X2
X5
ConvLSTM
Fully Connected Layer
Fully Connected Layer
Input size = (5,26,9)
filters=8, kernel size=5, padding=same, activation=tanh
filters=8, kernel size=5, padding=same, activation=tanh
units=16, activation=linear
units=9, activation=linear
Output size = (26,9)

Fig. 7. ConvLSTM-based prediction network architecture.

one dimension for space. Due to the 1D nature of the ECUs, we have used the 1D-ConvLSTM method in our proposal.

The ConvLSTM-based network's input represents the transferred CAN messages by all ECUs at a given timestep t, denoted by x_t in the shape of (#samples, #lookback, #ECU, #features). In this two-dimensional input data, each row maps to an ECU, while the columns are the features of the CAN payload transferred by the corresponding ECU. In a summary, the input to the prediction model can be considered as the sequential CAN features standing on a spatial matrix, with each spatial element being one ECU.

As can be seen in Fig. 7, the spatiotemporal prediction module consists of two 1D-ConvLSTM layers, each with eight filters of size 5. These ConvLSTM layers are responsible for extracting correlations between historical inputs from ECUs. Similar to LSTM, sequential patterns are learned through a gated structure, each controlling how much information to keep and forget. The filters are applied in each gate and pass through its input data. The learned weights in the filters demonstrate the amount of correlation within the data of neighboring ECUs. There are then two fully connected layers with 16 and 9 neurons, respectively, where the number of neurons in the last layer corresponds to the number of features to be predicted. The model is trained with Adam optimizer and mean absolute error as the loss function. An early stopping mechanism is utilized to prevent the overfitting of the model.

C. Comparison of LSTM-Based and ConvLSTM-Based Prediction Modules

As with the LSTM-based model, it is trained for each CAN ID separately. There are typically more than 100 electronic control units in a modern electric vehicle, all connected to one or more CAN buses. This implies having many CAN IDs in the network, and at the same time, many prediction models. The network will use more computation power and memory as a result of having to train more parameters. The ConvLSTM-GNB saves the overall IDS resources by sharing and reducing parameters. Although ConvLSTM has a smaller and power-saving model, the convolution operation used in its structure increases its computational time.

An LSTM unit is good at extracting temporal patterns. It learns and predicts the value of each ECU message independently, but it is not capable of considering the correlation between ECUs. Some of the sensors and their corresponding ECUs within the vehicle cooperate to perform functions, so it is more likely that their transmitted values are correlated, and the changes in the measured value of one ECU are affected by another. For instance, a vehicle's speed, RPM, engine temperature, and Carbon Dioxide emission values are all affected when it accelerates. However, not all ECUs on the CAN bus generate correlated data. For example, in the mentioned brake scenario, the measurements from the vehicle's internal temperature sensor are not affected. Uncorrelated signals play the role of noise for the ConvLSTM prediction model and cause the degradation of its performance. The localized models generated by LSTM-based modules focus on the signals of each CAN ID, without dealing with the noise of other ones.

Therefore, there is a tradeoff between the number of trainable parameters and the prediction performance. As an extreme example, if one ConvLSTM model is trained and deployed for input from all CAN IDs, fewer resources are consumed for training and maintaining the single model, but performance is lower due to uncorrelated signals interfering. Also, a single LSTM model for each CAN ID increases the memory and computational demands of the IDS; however, it avoids uncorrelated signals from being involved. It is important to consider the degree of correlation between CAN IDs in choosing the prediction module. One possible solution is to divide the CAN IDs into groups based on their interdependencies and apply a distinct ConvLSTM-based model. LSTM-based models can be used for individual CAN IDs that do not fit into the groups.

V. Performance Evaluation

This section discusses the experiments to test the performance of our proposed CAN network prediction-based IDS framework. The real dataset used is described first, followed by the evaluation procedure and the results.

A. Threat Model

ECUs send and receive messages on the shared CAN bus to communicate and operate the vehicle. Threats on CAN bus include disturbing the normal functions of vehicles by interrupting and tampering with the ECU messages. They can be viewed from different perspectives: target context and entering point.

CAN bus structure is not flexible, and the ECU nodes are not assumed to be added and removed from the bus. Hence, noticing a message with a new CAN ID other than what is already on the bus raises suspicions. Additionally, since ECUs transmit messages periodically or in response to events, deviations in the pattern of transmitted CAN IDs on the bus might indicate an attack. When messages with a higher priority (more dominant bits in the CAN ID field) are sent at a higher frequency than normal, they overload the bus and prevent other ECUs from accessing it. Another target for an attack is the message content. Attackers with escalated privileges can alter the content of CAN messages to forge an invalid measurement and modify the vehicle's behavior as a response to it.

Considering where attackers reside and how they obtain access to the network, we can classify them as insiders and outsiders. An insider has physical access to the bus, for example, by tapping into the OBD-II port. He can intercept and analyze messages as a first step, then, based on his understanding, impersonate another ECU and send messages on its behalf. The attacker can also be an outsider and take access to an ECU by compromising the remote wireless communication and causing it to malfunction.

To cover all the mentioned threats in our study, we have classified them based on the final impact on the CAN messages and designed our IDS solution to detect these types of attacks:

- DoS attack: This attack, also called the Flooding attack, aims at the availability of the CAN bus and tries to make it useless for transferring messages. The attacker uses the vulnerability of the access control mechanism in the CAN protocol. Using 0×000 as the highest priority in the CAN ID field, he sends forged messages with more dominant bits on the bus with high frequency. Because the wrongful messages are taking over the bus at all times, other ECUs are starved to the point where the legitimate messages cannot be transferred.
- Fuzzy attack: ECUs rely on CAN bus communication, which lacks authentication, making it difficult for them to distinguish between authentic and fraudulent messages. This vulnerability can be exploited through a fuzzing attack where an attacker examines the response of ECUs to fraudulent messages with random identifiers and contents. In a fuzzy attack, the attacker uses the CAN bus as a black box by randomly generating CAN frames to disrupt vehicle operations and discover meaningful messages.
- Spoofing attack: Spoofing attacks take advantage of the fact that the intruder knows the meaning of the CAN messages, whereas DoS and fuzzy attacks do not. Therefore, to achieve the attacker's specific goal, for example, changing the direction of the steer, the CAN payload content is modified to the invalid value. Since the result of the manipulated contents is the car's breakdown, it is also called a malfunction attack.

TABLE I
Summary of the Car Hacking Dataset

Attack type	Number of Messages		
	Total	Normal Messages	Injected Messages
Normal	988,987	988,987	0
DoS Attack	3,665,771	3,078,250	587,521
Fuzzy Attack	3,838,860	3,347,013	491,847
Gear Spoofing	4,443,142	3,845,890	597,252
RPM Spoofing	4,621,702	3,966,805	654,897

B. Dataset

To evaluate our proposed method, we use a publicly available dataset for the CAN bus offered by the Hacking and Countermeasure Research Lab (HCRL), The car Hacking Dataset [36]. It contains real-world CAN bus messages and attack scenarios on a variety of ECUs, making it useful for testing the robustness and effectiveness of security solutions in a real-world setting. The dataset consists of messages collected by connecting two Raspberry Pi3 devices to the OBD-II port of a real Hyundai YF Sonata car, one for logging the CAN traffic and one for injecting fabricated messages to the CAN bus. Four types of attacks, including DoS, fuzzy, RPM gauge, and gear spoofing, are implemented in this dataset. This dataset covers the widest attack range among all available datasets. For a DoS attack, high-priority CAN messages with an ID of 0×000 are fed every 0.3 milliseconds. In a Fuzzy attack scenario, messages with randomly spoofed IDs and Data are inserted every 0.5 milliseconds. The RPM/Gear spoofing attack targets a specific CAN ID and messages with spoofed contents are injected into the CAN bus every 1 millisecond. The messages in the data samples consist of a timestamp, CAN ID, DLC (Data Length Code), and Data [0-7] in hexadecimal format, and a flag indicating whether the sample is normal (R) or attack data (T). Moreover, this dataset provides bit representations of CAN messages, not decoded signals, making it a more general dataset than one specific to a manufacturer. A summary of the Car Hacking Dataset is given in Table I.

C. Results

The performance of the proposed IDS framework is tested by several experiments and compared to other state-of-the-art baselines. A computer with 16 GB RAM and an Intel® Core™ i7 processor running at 2.50 GHz is used to build and train the models implemented with the Keras framework and backend of Tensorflow. This level of computational power is

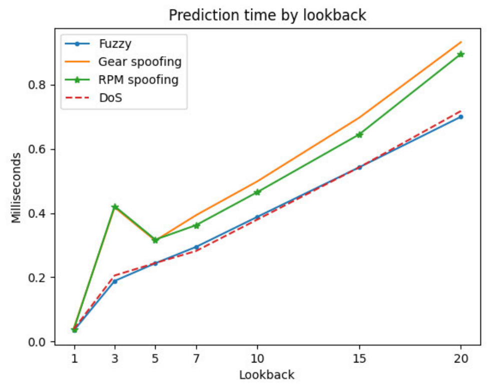


Fig. 8. ConvLSTM-GNB average response time per sample input for different attack types according to the size of historical observations.

similar to that of a typical onboard processing unit, such as the NVIDIA® Jetson™ AGX XAVIER, which consists of an 8-core Carmel ARM CPU operating at 2.26 GHz.

The dataset consists of normal and attack-related data files. We have trained and validated the prediction network using attack-free data. The other parts of the dataset are divided into two portions. The GNB is trained with twenty percent of the data, and the rest is utilized as test data to evaluate the detection performance of our method. The model's hyperparameters including lookback and network structure are tuned by the grid search method to achieve the highest performance.

Fig. 9 displays the F-score and accuracy of ConvLSTM-GNB IDS for various lookback values. Based on this figure, decreasing the lookback results in degraded IDS performance for RPM/Gear spoofing and DoS attacks. This is because the prediction module considers fewer historical observations and less information about the past to predict the upcoming message. Thus, it is unable to detect anomalies in time series data and cannot extract temporal correlations. Although, taking a long series of observations into account will reduce the detection performance as well because nearby elements are more likely to be correlated than those farther away. Also, Fig. 8 implies that longer sequence prediction results in longer processing time. To avoid slower detection and degradation of performance, the lookback value should be as small as possible. In the case of a Fuzzy attack, finding a suitable value for lookback seems to be challenging, since it has a different behavior compared to the other attack types.

The majority of CAN messages represent normal vehicle operation, while attacks on the CAN bus are rare. As a result, the CAN bus labeled data is imbalanced, with anomaly data representing the outlier minority. Traditional classification algorithms fail in the imbalanced data case and tend to bias toward the overabundant subclass. To address this challenge, One Class Classification (OCC) algorithms are used to deal with imbalanced data. The detection efficiency of our proposed approach is compared to some OCC methods as a baseline. The baseline methods are two one-class conventional classification methods, OCSVM (One-Class Support Vector Machines) with Stochastic Gradient Descent (SGD) and Isolation Forest, as well as a neural network-based method called LSTM Autoencoder. We also included the results of three existing works (Reduced Inception ResNet, NeuroCAN, and CANLite) discussed in Section II in the Comparison. The performance of the proposed LSTM-GNB and ConvLSTM-GNB IDS approaches was compared against these baselines and existing works in Table II to evaluate their effectiveness in detecting anomalies in the CAN bus data.

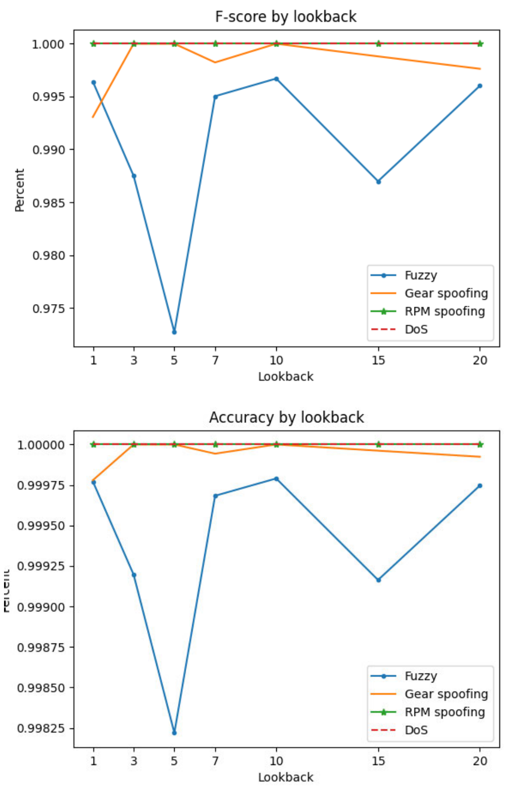


Fig. 9. ConvLSTM-GNB detection performance for different attack types according to the size of historical observations.

TABLE II

THE COMPARISON OF THE PROPOSED LSTM AND CONVLSTM-BASED IDS FRAMEWORKS WITH BASELINE AND EXISTING WORKS RESULTS ON THE CAR HACKING DATASET

Attack Types	Detection Methods	Recall	Precision	F-Score	Accuracy
DoS	OCSVM	1.000	0.835	0.910	0.968
	Isolation Forest	1.000	0.718	0.836	0.937
	LSTM-AE	1.000	1.000	1.000	1.000
	ReducedInception-ResNet [28]	0.9989	1.000	0.9995	0.9997
	NeuroCAN [29]	N/A	N/A	N/A	N/A
	CANLite [30]	N/A	N/A	N/A	N/A
	LSTM-GNB	1.000	1.000	1.000	1.000
	ConvLSTM-GNB	1.000	1.000	1.000	1.000
Fuzzy	OCSVM	0.987	0.793	0.879	0.965
	Isolation Forest	0.994	0.621	0.764	0.921
	LSTM-AE	0.998	0.977	0.987	0.989
	ReducedInception-ResNet [28]	0.9965	0.9995	0.9980	0.9982
	NeuroCAN [29]	N/A	N/A	N/A	N/A
	CANLite [30]	N/A	N/A	N/A	N/A
	LSTM-GNB	1.000	0.994	0.997	0.998
	ConvLSTM-GNB	0.997	0.996	0.997	1.000
Gear Spoofing	OCSVM	0 (TP=0)	0	-	0.833
	Isolation Forest	0 (TP=0)	0	-	0.835
	LSTM-AE	0.984	0.894	0.937	0.982
	ReducedInception-ResNet [28]	0.9989	0.9999	0.9994	0.9995
	NeuroCAN [29]	0.9063	0.9999	0.9508	0.8498
	CANLite [30]	0.9118	0.9999	0.9541	0.9792
	LSTM-GNB	1.000	1.000	1.000	1.000
	ConvLSTM-GNB	1.000	1.000	1.000	1.000
RPM Spoofing	OCSVM	0 (TP=0)	0	-	0.826
	Isolation Forest	0 (TP=0)	0	-	0.834
	LSTM-AE	0.995	0.893	0.987	0.982
	ReducedInception-ResNet [28]	0.9994	0.9999	0.9996	0.9997
	NeuroCAN [29]	1.000	1.000	1.000	1.000
	CANLite [30]	1.000	1.000	1.000	1.000
	LSTM-GNB	1.000	1.000	1.000	1.000
	ConvLSTM-GNB	1.000	1.000	1.000	1.000

TABLE III
THE COMPARISON OF DEPLOYMENT CONSIDERATIONS BETWEEN OUR PROPOSED PREDICTION MODULES AND EXISTING WORKS

Prediction Modules	#Model Parameters	Memory Footprint (KB)	Time to Predict (ms)	Model Input	Deployment Location
LSTM	5,673	95.7	0.06	Separate CAN IDs	Gateway
ConvLSTM	5,641	114	0.3	Correlated CAN IDs	Gateway
CLAM [32]	N/A	682	1.35	Correlated CAN IDs	ECU
TENET [31]	6,064	59.62	0.25	Separate CAN IDs	ECU
LSTM-P [24]	N/A	13,417	7.6	Separate CAN IDs	ECU

The results prove that our proposed methods outperform the baselines and existing works. Since anomalies that happen in the spoofing attack are group anomalies and not point anomalies, OCSVM and Isolation Forest fail to detect them. Individually, spoofed contents are in the range of normal data instances and they cannot be detected by considering them alone. However, they are anomalous when happening in the wrong position in a sequence of data points. Our time series-based prediction models possess the capability to consider every data point in a sequence, which enhances their anomaly detection performance. Additionally, the use of a fully data-driven Gaussian Naive Bayes approach to detect anomalies in the prediction errors, instead of relying on a predetermined threshold, further improves their detection accuracy.

The other observation from the results is that detecting fuzzy attacks is the most challenging one. In this type of attack, messages with random IDs and contents are fed into the network with high frequency. The high randomness confuses the prediction module and results in misidentifying some normal data points as anomalies, called false alarms. Even so, the false positive rate and precision are still lower than the baselines and above 99 percent.

ConvLSTM and LSTM prediction modules are both capable of extracting temporal features and detecting group anomalies in message flows. Furthermore, the results indicate that their detection performance is similar, outperforming other baselines. In the LSTM prediction module, one model is trained for each ECU independently, resulting in $26 \times 5,673 = 147,498$ trainable parameters in total. This large model is stored in the network, and distributed among ECUs. ECUs are responsible for checking each CAN message for normality by using their stored models. Therefore, the limited-resource ECUs have to deal with increased storage and computational demands.

The IDS with the LSTM prediction module is not an efficient structure. To avoid extracting similar temporal patterns multiple times, the separate models can be combined, and the model parameters can be reused. The ConvLSTM module accomplishes this by reducing the size of model parameters to 5,641 and predicting the message content of each ECU based on its historical messages and the observed content of other ECUs. Despite an increase in detection time caused by replacing multiplication with convolution operation, the ConvLSTM module's detection time is still acceptable. Table III compares the deployment considerations of LSTM and ConvLSTM prediction networks with those of existing works, in terms of the number of model parameters, memory footprint, prediction time, model inputs, and deployment location. Our proposed methods exhibit memory footprint and prediction time comparable to those of other works.

VI. CONCLUSION

In this paper, two anomaly-based IDS schemes are proposed for detecting attacks on the in-vehicle CAN bus to safeguard CAV, including LSTM-GNB IDS and ConvLSTM-GNB IDS. Both proposed solutions learn the usual variations of the CAN messages over time from the attack-free data. By having the last messages seen on the bus, they predict what message will come next, and then compare it with the actually observed one. The difference between expected and observed messages is computed with the mean absolute error (MAE) and the decision on the normality of the error is made by a GNB classifier. We have evaluated the detection performance of the proposed methods and compared them with three anomaly detectors, including two traditional machine learning methods OCSVM and Isolation forest, and one LSTM-based autoencoder. In contrast to baselines, LSTM-GNB and ConvLSTM-GNB show better performance, as they take into account the position of each message in a time sequence, rather than individually. Through consideration of the correlated time-series values generated by ECUs, ConvLSTM-GNB maintains almost the same level of performance while using significantly fewer resources than the LSTM-GNB model.

REFERENCES

[1] W. Feng, S. Lin, N. Zhang, G. Wang, B. Ai, and L. Cai, "Joint C-V2X based offloading and resource allocation in multi-tier vehicular edge computing system," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 2, pp. 432–445, Feb. 2023.
[2] D. Chen and Z. Lv, "Artificial intelligence enabled digital twins for training autonomous cars," *Internet Things Cyber-Phys. Syst.*, vol. 2, pp. 31–41, Jan. 2022.
[3] J. Huang, M. Zhao, Y. Zhou, and C. Xing, "In-vehicle networking: Protocols, challenges, and solutions," *IEEE Netw.*, vol. 33, no. 1, pp. 92–98, Jan. 2019.
[4] X. Sun, F. R. Yu, and P. Zhang, "A survey on cyber-security of connected and autonomous vehicles (CAVs)," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6240–6259, Jul. 2022.
[5] Q. Jiang, N. Zhang, J. Ni, J. Ma, X. Ma, and K. R. Choo, "Unified biometric privacy preserving three-factor authentication and key agreement for cloud-assisted autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9390–9401, Sep. 2020.
[6] K. Koscher et al., "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy*, May 2010, pp. 447–462.
[7] *Specification of Secure Onboard Communication*, Standard R20-11, AUTOSAR, Munich, Germany, 2020.
[8] A.-I. Radu and F. D. Garcia, "LeiA: A lightweight authentication protocol for CAN," in *Proc. 21st Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2016, pp. 283–300.
[9] Z. Lu et al., "LEAP: A lightweight encryption and authentication protocol for in-vehicle communications," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 1158–1164.
[10] C. Guo, Y. Wang, F. Chen, and Y. Ha, "Unified lightweight authenticated encryption for resource-constrained electronic control unit," in *Proc. 29th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Oct. 2022, pp. 1–4.
[11] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," in *Proc. World Congr. Ind. Control Syst. Secur. (WCICSS)*, Dec. 2015, pp. 45–49.

[12] M. Marchetti and D. Stabili, “Anomaly detection of CAN bus messages through analysis of ID sequences,” in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 1577–1583.

[13] M. Müter and N. Asaj, “Entropy-based anomaly detection for in-vehicle networks,” in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 1110–1115.

[14] A. Theissler, “Anomaly detection in recordings from in-vehicle networks,” *Big Data Appl.*, vol. 23, p. 26, Sep. 2014.

[15] M. Al-Saud, A. M. Eltamaly, M. A. Mohamed, and A. Kavousi-Fard, “An intelligent data-driven model to secure intravehicle communications based on machine learning,” *IEEE Trans. Ind. Electron.*, vol. 67, no. 6, pp. 5112–5119, Jun. 2020.

[16] O. Avatefipour et al., “An intelligent secured framework for cyberattack detection in electric vehicles’ CAN bus using machine learning,” *IEEE Access*, vol. 7, pp. 127580–127592, 2019.

[17] S. N. Narayanan, S. Mittal, and A. Joshi, “OBD_SecureAlert: An anomaly detection system for vehicles,” in *Proc. IEEE Int. Conf. Intell. Comput. Commun. (SMARTCOMP)*, May 2016, pp. 1–6.

[18] M. Levi, Y. Allouche, and A. Kontorovich, “Advanced analytics for connected car cybersecurity,” in *Proc. IEEE 87th Veh. Technol. Conf. (VTC Spring)*, Jun. 2018, pp. 1–7.

[19] V. S. Barletta, D. Caivano, A. Nannavecchia, and M. Scalera, “Intrusion detection for in-vehicle communication networks: An unsupervised Kohonen SOM approach,” *Future Internet*, vol. 12, no. 7, p. 119, Jul. 2020.

[20] D. Tian et al., “An intrusion detection system based on machine learning for can-bus,” in *Proc. Int. Conf. Ind. Netw. Intell. Syst.* Cham, Switzerland: Springer, 2017, pp. 285–294.

[21] M. Kang and J. Kang, “A novel intrusion detection method using deep neural network for in-vehicle network security,” in *Proc. IEEE 83rd Veh. Technol. Conf. (VTC Spring)*, May 2016, pp. 1–5.

[22] Y. Lin, C. Chen, F. Xiao, O. Avatefipour, K. Alsubhi, and A. Yunianta, “An evolutionary deep learning anomaly detection framework for in-vehicle networks–CAN bus,” *IEEE Trans. Ind. Appl.*, early access, Jul. 17, 2020, doi: 10.1109/TIA.2020.3009906.

[23] A. R. Javed, S. U. Rehman, M. U. Khan, M. Alazab, and T. Reddy, “CANintelliIDS: Detecting in-vehicle intrusion attacks on a controller area network using CNN and attention-based GRU,” *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1456–1466, Apr. 2021.

[24] 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. Anal. (DSAA)*, Oct. 2016, pp. 130–139.

[25] K. Zhu, Z. Chen, Y. Peng, and L. Zhang, “Mobile edge assisted literal multi-dimensional anomaly detection of in-vehicle network using LSTM,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4275–4284, May 2019.

[26] S. Longari, D. H. N. Valcarcel, M. Zago, M. Carminati, and S. Zanero, “CANnolo: An anomaly detection system based on LSTM autoencoders for controller area network,” *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1913–1924, Jun. 2021.

[27] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, “CANet: An unsupervised intrusion detection system for high dimensional CAN bus data,” *IEEE Access*, vol. 8, pp. 58194–58205, 2020.

[28] H. M. Song, J. Woo, and H. K. Kim, “In-vehicle network intrusion detection using deep convolutional neural network,” *Veh. Commun.*, vol. 21, Jan. 2020, Art. no. 100198.

[29] P. Balaji and M. Ghaderi, “NeuroCAN: Contextual anomaly detection in controller area networks,” in *Proc. IEEE Int. Smart Cities Conf. (ISC2)*, Sep. 2021, pp. 1–7.

[30] P. Balaji, M. Ghaderi, and H. Zhang, “CANLite: Anomaly detection in controller area networks with multitask learning,” in *Proc. IEEE 95th Veh. Technol. Conf., (VTC-Spring)*, Jun. 2022, pp. 1–5.

[31] S. V. Thiruloga, V. K. Kukkala, and S. Pasricha, “TENET: Temporal CNN with attention for anomaly detection in automotive cyber-physical systems,” in *Proc. 27th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2022, pp. 326–331.

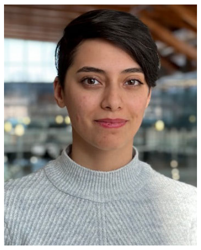
[32] H. Sun, M. Chen, J. Weng, Z. Liu, and G. Geng, “Anomaly detection for in-vehicle network using CNN-LSTM with attention mechanism,” *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10880–10893, Oct. 2021.

[33] J. Xiao, H. Wu, and X. Li, “Robust and self-evolving IDS for in-vehicle network by enabling spatiotemporal information,” in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun.; IEEE 17th Int. Conf. Smart City; IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Aug. 2019, pp. 1390–1397.

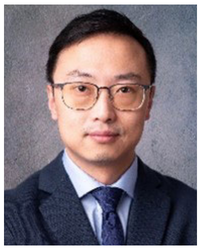
[34] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[35] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, “Convolutional LSTM network: A machine learning approach for precipitation nowcasting,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 802–810.

[36] 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.



Pegah Mansourian (Graduate Student Member, IEEE) received the bachelor’s degree in IT engineering from the University of Tehran and the master’s degree in computer networks from the Amirkabir University of Technology. He is currently pursuing the Ph.D. degree with the University of Windsor, conducting research in the field of connected autonomous vehicle security. His research interests encompass connected autonomous vehicles, cybersecurity, machine learning, and anomaly detection.



Ning Zhang (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Canada. He is currently an Associate Professor and the Canada Research Chair of the Department of Electrical and Computer Engineering, University of Windsor. His research interests include connected vehicles, mobile edge computing, wireless networking, and security. He is also a Distinguished Lecturer of IEEE ComSoc, a Highly Cited Researcher (Web of Science), and the Vice Chair of IEEE Technical Committee on Cognitive Networks and IEEE Technical Committee on Big Data. He serves/served as an Associate Editor for IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, IEEE INTERNET OF THINGS JOURNAL, and IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING. He also serves/served as the TPC/General Chair for numerous conferences and workshops, such as IEEE ICC, VTC, INFOCOM Workshop, and Mobicom Workshop.

Arunita Jaekel received the B.Eng. degree in electronics and telecommunications engineering from Jadavpur University, Kolkata, India, and the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Windsor, Windsor, ON, Canada. Since 1995, she has been a Faculty Member with the School of Computer Science, University of Windsor, where she is currently a tenured Professor. Her research is supported by grants from the Natural Sciences and Engineering Research Council (NSERC), Canada. She has authored or coauthored more than 100 refereed articles. Her current research interests include secure vehicle-to-vehicle communication, congestion control, and the design of reliable wireless sensor networks.

Marc Kneppers received the master’s degree in astronomy from the University of Western Ontario (Western University), Canada. He has 20 years of experience in IT/networking security and was appointed a TELUS Fellow. He is currently a Chief Security Architect with TELUS Communications. His responsibilities include security oversight and strategy across all of TELUS technologies and portfolios. He represents TELUS on Canadian national infrastructure forums and industry boards with membership in international security forums and vendor advisory boards.