MULSAM: Multidimensional Attention With Hardware Acceleration for Efficient Intrusion Detection on Vehicular CAN Bus

He Xu, Xiaokang Shi, Hansheng Liu, Yanwen Wang, *Member, IEEE*, Jiwu Lu, *Member, IEEE*, Haibo Zeng, *Member, IEEE*, Renfa Li, *Senior Member, IEEE*, and Di Wu, *Member, IEEE*

***Abstract*—Controller area network (CAN) protocol is an efficient standard enabling communication among electronic control units (ECUs). However, the CAN bus is vulnerable to malicious attacks because of a lack of defense features. In this article, a novel vehicle intrusion detection system (IDS) is developed. The challenge is that existing techniques of IDSs rarely consider attacks with small-batch, which are characterized by their small attack scale and concealed attack patterns, posing a significant threat to driving safety. To solve this problem, we developed an algorithm model that merges multidimensional long short-term memory (MD-LSTM) and self-attention mechanism (SAM), shortly named MULSAM. The MULSAM model was compared with other baseline models, including stacked long short-term memory (LSTM), MD-LSTM, etc. Experiments show that our approach has the best-detection accuracy (98.98%) and training stability. Further, to speed up the inference of MULSAM on edge, the hardware accelerator is implemented on FPGA devices using technologies, such as parallelization, modular, pipeline, and fixed-point quantization. Experiments show that our FPGA-based acceleration scheme has a better-energy efficiency than the CPU platform. Even with a certain degree of quantification, the acceleration model for MULSAM still displays a high-detection accuracy of 98.81% and a low latency of 1.88 ms.**

***Index Terms*—Abnormal detection, controller area network (CAN), FPGA, multidimensional long short-term memory (LSTM), self-attention mechanism (SAM).**

Received 7 May 2024; revised 6 December 2024; accepted 2 February 2025. Date of publication 12 February 2025; date of current version 22 August 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 61932010 and Grant 61972145; in part by the Science and Technology Innovation Program of Hunan Province under Grant 2024RC3105; and in part by the Shenzhen Science and Technology Program under Grant JCYJ20240813162405008. This article was recommended by Associate Editor Y. Jin. *(He Xu and Xiaokang Shi contributed equally to this work.) (Corresponding authors: Di Wu; Yanwen Wang; Jiwu Lu.)*

He Xu, Hansheng Liu, Jiwu Lu, and Di Wu are with the National Engineering Research Center for Robot Visual Perception and Control Technology, Hunan University, Changsha 410082, Hunan, China (e-mail: xuhe@hnu.edu.cn; z415663988@hnu.edu.cn; jiwulu@hnu.edu.cn; dwu@hnu.edu.cn).

Xiaokang Shi and Yanwen Wang are with the College of Electrical and Information Engineering, Hunan University, Changsha 410082, Hunan, China, and also with Shenzhen Research Institute, Hunan University, Shenzhen 518000, China (e-mail: shixiaokang2022@hnu.edu.cn; wangyw@hnu.edu.cn).

Haibo Zeng is with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA (e-mail: hbzeng@vt.edu).

Renfa Li is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, Hunan, China (e-mail: lirenfa@hnu.edu.cn).

Digital Object Identifier 10.1109/TCAD.2025.3541566

I. Introduction

CONTROLLER area network (CAN) bus protocol has been widely used in industrial automation control due to its low cost, high reliability, real-time, and robust anti-interference ability [1]. In effect, the CAN bus has become a communication standard in the automotive field [2]. The electronic control units (ECUs) perform identity authentication through the CAN ID of the CAN data frame on the CAN bus [3]. The majority of ECUs in vehicles are interconnected via the CAN bus for data exchange, including common types of in-vehicle communication networks, such as the engine control module (ECM), transmission control module (TCM), and body control module (BCM). Therefore, CAN communication protocol has become one of the most promising and widely used network technologies in automotive systems thanks to its high reliability, real-time performance, and low-cost properties. However, CAN ID can be arbitrarily modified, which allows intruders to attack the network [4]. For example, the intruder can launch a DoS attack to preempt the data transmission window time of the CAN network, making other legitimate ECUs fail to work continuously and may even result in the bus-off, which refers to a specific error state indicating that a node has been forced to disconnect from the CAN bus due to detecting an excessive number of errors in the CAN network. Moreover, small-scale attacks, such as replay and delete attacks, do not require frequent injections to attack [5], [6]. Under such attacks, the distribution of CAN IDs is similar to that of normal CAN IDs, making them difficult to detect and potentially posing greater risks that could lead to more severe consequences. For example, an attacker could use a replay attack to resend modified speeds, causing a vehicle to operate at unsafe speeds and disrupting normal driving. Alternatively, a delete attack could prevent a specific ECU unit from transmitting data to the CAN network, resulting in accidents or other safety issues. In short, due to the communication characteristics of the CAN protocol, the CAN bus network has inevitable safety hazards, such as illegal control, data leakage, and so on.

A. Motivations

Intrusion detection technology is widely used on the in-vehicle CAN bus. In recent years, with the growth of intelligent connected vehicles (ICVs) [7], more and more

attacks are targeting the in-vehicle system, especially the elaborate-designed attacks with small-batch characteristics that are extremely deceptive and destructive. However, few researchers have designed deep learning models for attacks with small-batch characteristics and practical deployment on embedded systems. Inspired by the multidimensions of RoseTTAFold [8], which has excellent performance on protein structure prediction, multidimensional means are used to design our intrusion detection system (IDS) based on multidimensional long short-term memory (MD-LSTM), which can deploy long short-term memory (LSTM) cells along any or all of the dimensions.

To deploy our model on the feasible FPGA platform, our IDS is specially designed and created in two dimensions [9]. At the same time, to compensate for the loss caused by the reduction of dimensions, the fusion of the self-attention mechanism (SAM) [10] is utilized to improve the detection performance. The MD-LSTM with SAM, called MULSAM, can learn multidimensional features of CAN time-series data, providing agile and stable processing at the CAN bus network edge. Due to the effect of the self–attention mechanism, MULSAM can better-separate data, which has more complex features and more separated interdependencies than standard MD-LSTM networks.

B. Challenges

With the unceasing improvement of the automotive intelligence level, the number of in-vehicle network ECUs has been gradually increasing, which makes the in-vehicle network more complex. Unfortunately, the CAN bus lacks an effective security mechanism to resist external intrusion attacks [11], [12]. The exposed interfaces, such as GPS, V2V, 4G/5G, and so on, have imported many unpredictable security threats to the automobile [13], [14]. Also, with the wireless V2X connection, attackers have more opportunities to access the vehicle network to obtain vehicle information and even remotely control the vehicle. The original built-in safety mechanism of the CAN bus is mainly to ensure reliable communication. However, intrusion attacks on the CAN bus now can cause malfunction, jam, and data tampering of the vehicle network communication. These eventually cause abnormal vehicle driving conditions, which endangers the safety of vehicles and drivers. It may also involve personal privacy data leakage problems and lead to property damage. Therefore, the security defense methods of communication systems are becoming more and more critical.

Today, the CAN bus protocol plays an essential role in the in-vehicle electronic system. Any abnormal information transmission caused by intrusion attacks may cause abnormal working status and endanger the vehicle's safe driving, causing unpredictable loss and damage. Therefore, detecting abnormal data transmission quickly and efficiently on the CAN bus in ICVs is crucial and important [15]. Through the intrusion detecting technology, the vehicle generates an alarm message and switches into a safe protection mode [16]. However, the widely used automotive embedded systems have limited hardware computing resources due to cost constraints. If the IDS is directly deployed into the in-vehicle system, it will have a performance tradeoff on the vehicle system itself. Therefore, most researchers plug IDS hardware externally onto the CAN bus network and conduct intrusion detection experiments by monitoring CAN bus data transmission messages [17], [18]. The advantage of this method is that no change in the hardware architecture is needed. However, to ensure the safety of vehicles, IDSs are compulsory to be real-time and efficient, which is difficult for the existing automotive embedded system. Therefore, although challenging, it is worthwhile to implement an IDS in the vehicle, where there are few related studies as far as we know.

C. Our Contributions

In this work, a compact, novel deep learning model is proposed, which is based on the MD-LSTM network with a self-attention layer, aiming to improve the performance of IDS under multiple different attacks on the vehicle's CAN bus. At the same time, the model reconstruction and FPGA-embedded platform implementation scheme are explored in this article. An efficient and practical solution is provided for designing IDSs in ICVs.

Our critical contributions are summarized as follows.

1) We use the attack-free data from an actual car running on the road, but there is a lack of real-world data for attacks. To address this challenge, we build a simulation system to generate attack datasets, including those for DoS, fuzzy, spoofing, replay, and delete attacks. The design of the simulation system is based on analyzing the CAN ID distribution of these common attack types. Our observation is that the time-series data of CAN message IDs is correlated to the function of the in-vehicle system. For example, after an ECU sends a message over the CAN network, the receiving ECU will only be able to process and possibly respond by sending another message (with a different ID) after a certain amount of delay. Usually, this dependency makes the distribution of CAN IDs in a relatively stable state.
2) The MULSAM is developed to analyze large volumes of real-time CAN data and optimize network performance. The multidimensional and SAMs are adopted to make the MULSAM tiny and parallel, which is suitable for deployment on an FPGA-embedded device. The role of the self-attention layer is to convert the input data into an intermediate semantic representation, making its characteristic information more evident and easy to distinguish, which can be regarded as an encoding process. It can enhance MD-LSTM cells' depth and temporal computation, which is capable of processing multidimensional CAN data with attack features. The safety of networked systems can be improved by real-time, efficient processing at the edge device in the vehicle. New IDS based on the data flow and learning driven paradigm can perform better by harnessing the real-time CAN data.
3) The Stacked LSTM, MD-LSTM, and our MULSAM model are designed and implemented on the FPGA-embedded device (Ultra96-V2 board). The

FPGA-embedded platform, which has the advantage of parallel processing and the ability to customize hardware algorithms, can quickly and flexibly implement our deep learning model. It is appropriate for application scenarios that require high-real-time performance. In this article, the matrix multiplication operation of the dynamic matrix is presented for the SAM computation to improve the internal throughput of the MULSAM model. A data flow-based design paradigm is also provided for the FPGA-based implementation of the MD-LSTM cell. Experiments show that FPGA-based MULSAM has a higher-energy efficiency than the CPU platform. Compared with other LSTM-related deep learning models, it also has a higher-detection accuracy and lower latency.

II. Related Works

From the perspective of detection approaches, the design of IDS can be divided into rule-based and machine learning (ML)-based categories.

A. Rule-Based IDS

The rule-based IDS is a simple system, which uses some internal logic relationships of CAN data to perform anomaly analysis. For instance, Hoppe et al. [19] studied the regularity of data sent by a specific ECU in the CAN bus and proposed an IDS based on the strategy of abnormal signals. However, this method has great limitations and poor flexibility because it needs to study the data characteristics of the ECUs in depth. Vuong et al. [20] designed an attack detection method based on decision trees for cyber–physical systems and evaluated the model against various scenarios involving DoS, command injection, and two malware attacks. However, the proposed method approximately has a detection latency of 1s, which is too large for ICVs. Ling and Feng [21] combined the CAN IDs with their occurrence frequency and counted the number of CAN messages that belong to the given CAN ID for detecting malicious CAN messages. Although it is simple, the proposed algorithm has limited capability to detect attacks with small-batch. Cho and Shin [22] extracted and estimated transmitters clock skews, which are fingerprints of transmitters' ECUs, to solve the linear parameter identification problem in IDS. However, this method can detect attacks only for periodic messages and the attacker can manipulate the frame data to bypass it [23].

With the increasing complexity of vehicular functions, the number of ECUs has gradually increased, and the in-vehicle communication network has become more complex and volatile. It is hard to meet the development of communication security in the automotive by decision making with simple rules.

B. Machine Learning-Based IDS

The ML-based IDS is able to process more complex data with multidimensional features, but it is also harder to train than rule-based IDS besides posing higher-deployment cost. BTMonitor [24] extracted nine essential features in the time and frequency domain as the fingerprint features of ECUs and completed classification tasks to identify intrusive ECUs through the multilayer perceptron (MLP). VoltageIDS [23] extracted the essential features from the message signal and used the multiclass classifier to classify the CAN ID of the message. It also can distinguish between errors and bus-off attacks. Unfortunately, for both BTMonitor and VoltageIDS, the temperature and the electromagnetic environment significantly influence the detection accuracy. CANet [17] proposed an unsupervised learning approach that combined LSTM and autoencoder to train the time series of CAN messages. However, the independent LSTM input model for each ID in CANet is complicated and hard to deploy in the in-vehicle system. Hossain et al. [18] generated three attack datasets based on the attack-free traffic from a real car, and proposed an LSTM-based IDS to detect and mitigate the CAN bus network attacks, including DoS, fuzzy, and spoofing attacks, but did not consider the delete attack that disables a vehicular function, which is common in vehicle bus-attack.

Since the vehicular electronic system is in a mobile state during driving, the power consumption and the detection of small-batch attacks are vital issues for ML-based IDS. Unfortunately, there are few related works exploring embedded implementation and addressing small-batch attacks. Correspondingly, we present an enhanced ML-based IDS with efficient design and implementation on the FPGA platform for the actual vehicular system.

III. Background

A. Self-Attention Mechanism

The SAM is an advanced technique designed for modeling intricate interdependencies among elements in sequential data. Its core principle involves treating each element of the input sequence as a query (Q), a key (K), or a value (V), facilitating an in-depth computation of similarity between queries and keys, which translates into significance scores for interrelations within the sequence. Subsequently, these scores are weighted and aggregated as the output.

The operation steps start with the computation of queries, keys, and values for each element within the input sequence through linear transformations of the input data, followed by partitioning into numerous 'heads' for parallel computing. Attention scores are then derived by evaluating the similarity (e.g., via dot products or other similarity measures) between each query and all corresponding keys across the sequence, thus yielding a matrix of attention scores that highlight the relative importance of sequence positions about one another. By applying a softmax normalization to the attention scores, one obtains a set of attention weights that embody a probabilistic distribution reflecting the weighted significance across the sequence. Finally, attention weights are used for a weighted summation to achieve a final output that integrates the influence of all sequence positions. This weighted summation ensures that the output represents a dynamic, contextually informed weighted mean of the values, optimized by the attention mechanism.

The SAM is superior in its ability to directly capture long-distance dependencies within a sequence without

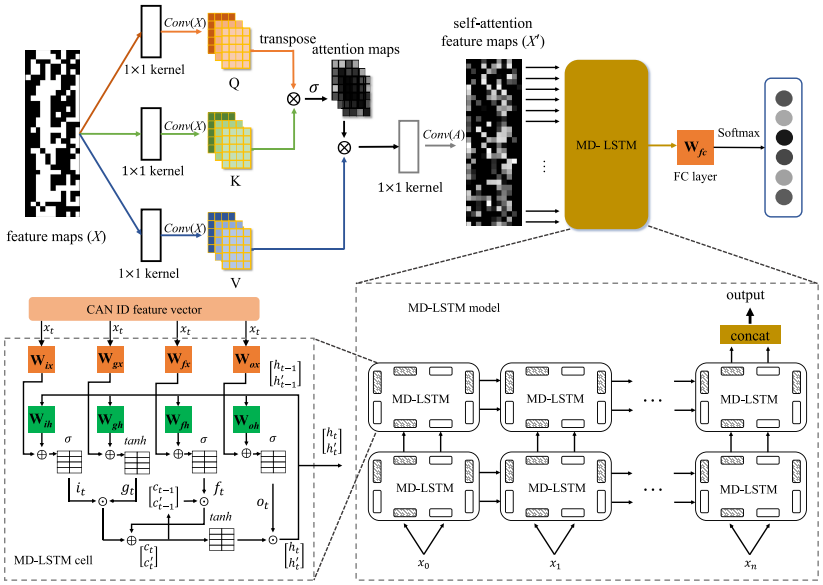


Fig. 1. Architecture of MULSAM.

any dependence on recursive or convolutional architectures, achieving impressive results in processing sequence data, such as natural language data.

B. MD-LSTM

MD-LSTM is an enhanced LSTM model, primarily improving the performance of LSTM by innovatively incorporating a multiscale mechanism. This advanced design allows the model to adeptly handle informational variations spanning a diverse array of temporal scales, thereby significantly improving its proficiency in capturing long-term dependencies contained within sequential datasets.

MD-LSTM introduces two pivotal concepts: 1) multiple-dilation and 2) multiple-layer configurations. The multiple-dilation involves a composite of several LSTM units each with distinct dilation coefficients, comprising a multiple-dilation network architecture. The dilation coefficient of each LSTM unit dictates the temporal stride spanned across the sequence. A larger coefficient enables the unit to capture more extended long-term dependencies. Multiple-layer in MD-LSTM typically refers to the stacking of several multiscale LSTM layers, which serves to enhance the deep representation capability of the model. Such a layered configuration allows the model to encapsulate features across various degrees of abstraction within the sequential data.

Training of the MD-LSTM follows procedures similar to the LSTM model, employing optimization methodologies like backpropagation and stochastic gradient descent for end-to-end training. MD-LSTM has manifested outstanding outcomes in various fields, including video analysis and natural language processing, particularly superior in processing long sequence data and capturing long-term dependencies within sequences.

IV. Overview of MULSAM System

Given the advantage of LSTM to track temporal dependencies, a tiny, novel system is proposed, which adapts its benefits and extends its structure by MD-LSTM specifically to process the CAN time-series data. Moreover, a SAM is added to strengthen the correlation between time series data. As shown in Fig. 1, the input data is first transformed into the processed data after the self-attention layer in the proposed system. Then the processed data is transmitted to MD-LSTM in time steps to obtain the output of hidden layers in two dimensions. Third, the outputs of MD-LSTM are concatenated to fuse distinct features of different dimensions. Finally, the classification result is obtained through a fully connected layer and the Softmax activation function.

From Fig. 1, it is essential to notice that the MULSAM proposed in this article utilizes the SAM to make up for the loss caused by reducing the dimension of the MD-LSTM. Thus, MULSAM can more effectively process the time series of CAN data with multidimensional features. Therefore, for anomaly detection in the CAN bus network, the accuracy of the MULSAM model is consistently better than other ML models.

V. Attack Model and Datasets

This article focuses on five types of attacks: DoS, fuzzy, spoofing, replay, and delete. The DoS, fuzzy, and spoofing attacks have the characteristic of flooding injection, while replay and delete attacks have the attribute of small-batch. Compared to flooding attacks, small-batch attacks have much less impact on the network payload and are much harder to detect.

DoS Attack: Once attackers invade the CAN bus network, and will continue to send the highest-priority CAN data frame to the CAN bus network, thus occupying the data transmission time window of other normal ECUs. This results in the CAN bus being in a congested state, which may cause the in-vehicle system to be paralyzed, and endanger the safety of the vehicle.

Fuzzy Attack: The attackers can quickly launch the fuzzy attack without knowing the specific information of the ECUs

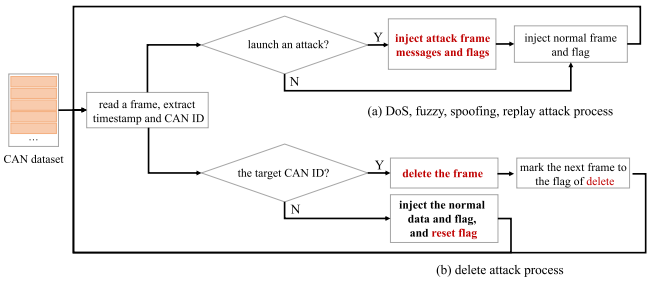


Fig. 2. Generate process of five attacks.

on the CAN bus of the vehicle system. The difference from the DoS attack is that fuzzy attack injects randomness data. Because of the randomness of CAN IDs and the characteristics of mass injection, it has a certain probability of coinciding with the ECU' CAN ID existing in the vehicle network. In this case, it will be possible to deceive the vehicle system.

Spoofing Attack: In a spoofing attack, an intruder listens to the CAN bus but does not decipher the CAN bus function. It eavesdrops on CAN messages and then injects many of the same CAN data, which drives the ECUs to receive outdated messages and misjudge.

Replay Attack: The intruder can listen to the CAN bus and decipher the CAN signal, which enables more precise replay attacks, such as accelerations or changes in driving direction.

Delete Attack: When an intruder invades the CAN bus network, it may cause a legitimate, important ECU to lose the function of sending data to the CAN bus network. In this case, the CAN ID corresponding to the ECU will not appear in the CAN bus network, which is called the delete attack.

A. Generation of Attack

This article uses the open-source CAN bus dataset from the 4TU.ResearchData (an international data repository for science, engineering, and design) [25]. The dataset was collected from the actual car while driving. The five types of attack CAN data are generated as shown in Fig. 2. Generating attack data requires precise timing and selection of target IDs to determine whether to launch a specific type of attack. Specifically, for DoS, fuzzy, spoofing, and replay attacks, attack intervals and the values of the target CAN IDs should be preset before generating the attack data. Then the CAN data from the original dataset are extracted, in which we use the data's timestamp along with the preset intervals to determine when to initiate the attack. Once an attack is determined to be launched, we inject the predetermined attack ID and mark the status at the corresponding interval to simulate different types of attacks. For delete attacks, we first set a target ID value, then continually extract CAN data from the original dataset. Next, we decide whether the current CAN ID is the target ID for the attack. If it is the target ID, we delete the current CAN ID and mark the next frame of the CAN message as a delete attack marker, thereby simulating a delete attack. Through the above attack methods, this article generates five anomalous datasets with attack characteristics. Fig. 3 shows the distribution of CAN ID characteristics in the normal state and in five attack datasets.

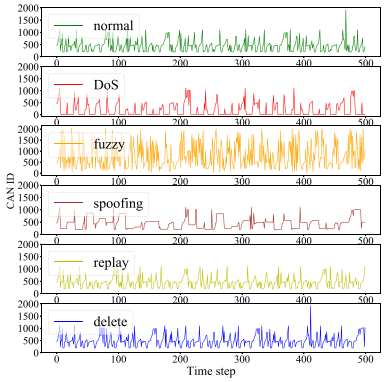


Fig. 3. Distribution of attacks' CAN ID.

From Fig. 3, the difference between the characteristics of DoS, fuzzy, and spoofing attacks with the normal data state can be seen clearly due to their flooding features. However, the characteristics of replay and delete attacks are not evident, which means that small-batch attacks is more difficult to distinguish because it is similar to the benign state. The experiment in Section VII also proves this assumption.

B. Preprocessing

The input data usually needs to be normalized to speed up the training network fit in the deep learning training progression. If the CAN ID is directly used as the input feature, it will cause the accuracy of the input data to decrease because the FPGA design needs to perform quantization processing. To avoid this situation, we convert the CAN ID to the bit features and use 0 or 1 as the input feature.

First, the original CAN ID is expressed in a hexadecimal system, which occupies 2 bytes, where only the lower-11 bits are valid. Equation (1) can calculate the 11-bit features in the original CAN ID data

$$x_i = \begin{cases} 0, \ can\_id \ \& \ (1 << i) = 0 \\ 1, \ can\_id \ \& \ (1 << i) > 0 \end{cases} i \in (0, 1, \ldots, 10) \quad (1)$$

where *can_id* represents the original CAN ID, i represents the bit position to be extracted, x_i is the ith bit value, & represents bitwise AND operation, and $<<$ represents left shift operation. By extracting the bit feature of the CAN ID as the input series, normalizing the input value can be avoided. Since the input value is only 0 or 1, we can use 1-bit data wide for storage in an FPGA device, which can greatly reduce the resource overhead without extra computing consumption.

The distribution characteristics after converting the CAN ID integer into 11-bit data are shown in Fig. 4, where the x-axis is the corresponding position of the bit, and the y-axis is the corresponding time step.

VI. MULSAM Design

This article proposes a MD-LSTM architecture with the front SAM (MULSAM). The MULSAM model can be divided into two primary parts, including a self-attention layer and a MD-LSTM network. First, the self-attention layer, which is widely used in the transformer model, is utilized to enhance the correlation between the time-series data and easily

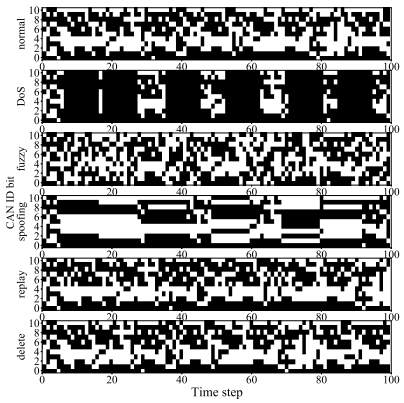


Fig. 4. Distribution of attacks' CAN ID bits.

distinguish attack features. The self-attention layer can reduce dependence on external information and better capture the internal correlation of features. Second, the rear part of the model is a MD-LSTM network, which can extract deeper characteristics from the time-series data.

Attacks with small-batch are more difficult to detect due to their lower-attack frequency. The SAM can assign higher weights to key parts of a sequence, making it more sensitive to detecting local anomaly patterns within sequences. However, the SAM may lack a comprehensive understanding of time series data, performing inadequately in scenarios where anomalies are not clearly related to the overall characteristics of the sequence. On the other hand, MD-LSTM, with its long-term memory capability, can continuously track contextual information across the entire sequence, making it particularly effective in understanding the overall characteristics of the sequence under attack. However, MD-LSTM is less effective at capturing anomalies at specific time points. Combining self-attention and MD-LSTM leverages the efficient ability of the SAM to focus on local information within the sequence, while the MD-LSTM can provide a robust capability for understanding global information in sequence. Therefore, the integrated model is more sensitive to small-scale attacks.

A. Neural Network Design

1) Self-Attention Layer in MULSAM: The intention of the self-attention layer can be considered as preprocessing the input data by allocating different weight values. As shown in Fig. 5 (left), the typical architecture of the self-attention layer uses a fully connected network to get the internal Q, K, and V matrices and the Softmax function as the internal activation function. To simplify the internal calculation of the self-attention layer and slash the volume of network parameters, the fully connected network is replaced with the convolutional network, and the Softmax activation function is modified to the logic sigmoid activation function as shown in Fig. 5 (right). This scheme also benefits the model algorithm design based on the FPGA platform to execute better parallelly.

First, the input data is defined as follows:

$$X = (x_0, x_1, x_2, \ldots, x_n). \tag{2}$$

where X is the input vector, n is the time step of one input data, and x_n represents a CAN ID with 11 dimensions.

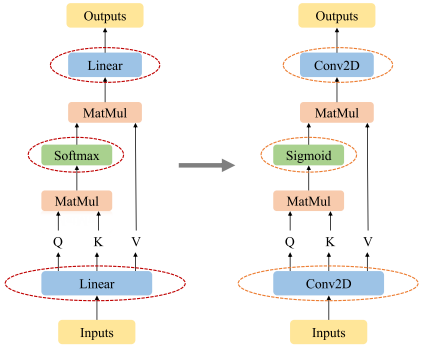


Fig. 5. Improvement of self-attention.

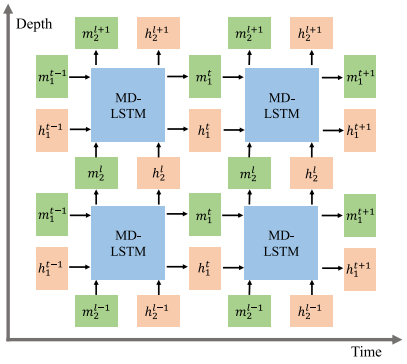


Fig. 6. Architecture of MD-LSTM.

The internal calculation process of the self-attention layer can be expressed as follows:

$$(Q, K, V) = \text{Conv2D}(X, (1, 1), \text{Hidden\_Dim} * 3) \tag{3}$$

$$\tilde{A} = \sigma\left(KQ^T\right) \tag{4}$$

$$\text{Attn}\left(\tilde{A}, V\right) = \text{Conv2D}\left(\tilde{A}V, (1, 1), \text{Output\_Dim}\right) \tag{5}$$

where the *Hidden_Dim* stands for the number of output channels of an internal matrix (Q, K, or V). The value of the output matrix of (3) is dimensional evenly divided into three parts, namely, Q, K, and V. Then, the attention maps are calculated after activation through (4), and the output of the self-attention layer is obtained according to (5). The *Output_Dim* represents the number of final output channels, and the size of the input will be equal to that of the output when *Output_Dim* is set to 1.

2) MD-LSTM in MULSAM: Unlike the traditional stacked LSTM, the MD-LSTM network [9], as shown in Fig. 6, adds LSTM cells along the depth dimension and the temporal dimension of the network. This architecture gives the depth dimension the same gradient channeling properties available along the temporal dimension, which mitigates the vanishing gradient problem in networks and extract deeper features.

The weight of different dimensions in the MD-LSTM network can be individual or be shared in the storage. When the different dimensions are independent, the calculation process of each dimension can be parallel, which is beneficial to the performance of the FPGA device. Thus, the design strategy of independent dimension is followed in this article, and the MD-LSTM cell can be split into two LSTM cells.

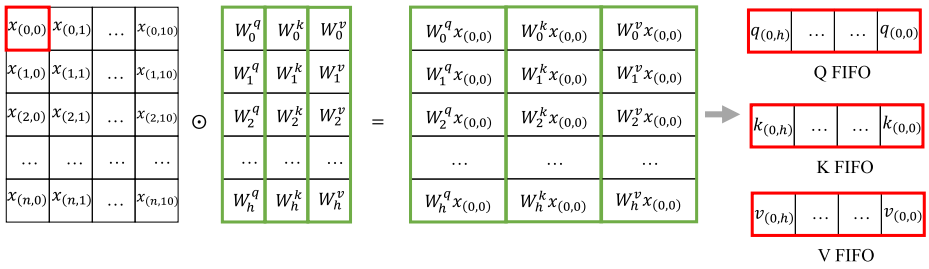


Fig. 7. Step computation of (Q, K, V) calculation module. n is $(N-1)$ and h is $(H-1)$.

Each LSTM cell in a different dimension uses a hidden state together with a memory cell to communicate with the next. The computation of the LSTM cell at each step is updated as follows:

$$\begin{aligned} g_t &= \tanh\left(W_{gx}x_t + W_{gh}h_{t-1} + b_g\right) \\ i_t &= \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + b_i\right) \\ f_t &= \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right) \\ o_t &= \sigma\left(W_{ox}x_t + W_{oh}h_{t-1} + b_o\right) \\ c_t &= g_t \odot i_t + c_{t-1} \odot f_t \\ h_t &= \tanh(c_t) \odot o_t \end{aligned} \tag{6}$$

where σ is the sigmoid function, $W_{gx}, W_{ix}, W_{fx}, W_{ox}$ are the recurrent weight matrices of the input vector, and $W_{gh}, W_{ih}, W_{fh}, W_{oh}$ are the recurrent weight matrices of the hidden vector. The functional LSTM$(\cdot,\cdot,\cdot,\cdot,\cdot)$ is used as shorthand for (6) as follows:

$$(h_t, c_t) = \text{LSTM}(x_t, h_{t-1}, c_{t-1}, W_i, W_h). \tag{7}$$

Unlike the computation of LSTM, a MD-LSTM block receives an input of two hidden vectors and two memory vectors from the depth and temporal dimensions. The computation is concise and proceeds as follows:

$$\begin{aligned} \left(h_t^1, c_t^1\right) &= \text{LSTM}\left(x_t, h_{t-1}^1, c_{t-1}^1, W_i^1, W_h^1\right) \\ \left(h_t^2, c_t^2\right) &= \text{LSTM}'\left(x_t, h_{t-1}^2, c_{t-1}^2, W_i^2, W_h^2\right). \end{aligned} \tag{8}$$

Each dimension has different weight matrices that correspond to the standard LSTM mechanism. Then these output hidden vectors are concatenated into a new vector H as the final output vector of MD-LSTM as follows:

$$H = \left[h_N^1, h_N^2\right] \tag{9}$$

where N in (9) is the total number of time steps. In practice, a one-time step indicates the time required for the system to generate one CAN data message.

B. FPGA-based Model Design

How to improve the data locality of matrix structures is a crucial problem for maximizing the performance of the ML model. An automated caching mechanism is used to improve the data locality in CPUs and GPUs, while FPGAs allow the developer to allocate data structure resources [26]. To implement MULSAM application deployment at the edge device, we analyze the internal parallelism of the algorithm,

Algorithm 1 (Q,K,V) Calculation

Input: time series of CAN ID (X)
1: **for** $x[i]$ in X **do**
2: **for** $j = 0$ to 10 **do**
3: **if** $x[i]$ & $(1 << j) = 0$ **then**
4: $b[i \times n + j] = 0$
5: **else**
6: $b[i \times n + j] = 1$
7: **end if**
8: **for** h = 0 to (H - 1) **do**
9: $q[h][i \times n + j] = b[i \times n + j] \times W_h^q$
10: $k[h][i \times n + j] = b[i \times n + j] \times W_h^k$
11: $v[h][i \times n + j] = b[i \times n + j] \times W_h^v$
12: **end for**
13: **end for**
14: **end for**
Output: Q, K, V

and the hardware circuit is realized by Vivado high-level synthesis (HLS). As the Neural Network design above, the FPGA-based network design is also split into two parts, including the self-attention pipeline design and MD-LSTM pipeline design. By connecting each independent calculation module through the FIFO resources, the whole calculation process can be streamlined.

1) Self-Attention Pipeline Design: For CPUs or GPUs, each step of the computation of the self-attention layer, as shown in Fig. 5 (right), requires waiting for the completion of the previous step. For example, The KQ^T operation in (5) can be computed when the Q and K matrices are fully completed. The algorithm needs to be refactored because each module's input and output stream are FIFO queues instead of a complete matrix, as shown in Fig. 7.

(Q, K, V) Calculation Module Design: The Q, K, and V matrices inside the self-attention layer do not have temporal interdependency, so they can be combined and calculated simultaneously, no matter whether it is in the CPU, GPU, or FPGA architecture. Thus, a module is built for the internal calculations of Q, K, and V, where the input stream data is a time series of CAN ID bit features, and the output streams are Q, K, and V streams transformed from the matrices. Algorithm 1 shows the whole process of this module.

Activation Calculation Module Design: Since the Q, and K matrices are calculated sequentially, then KQ^T in (5) cannot be fully calculated at one time. The calculation expression of the element of KQ^T is as follows:

$$a_{i,m} = \sum_{j=0}^{11\times N-1} k_{i,j} q_{j,m} i, m \in (0, 1, \ldots, H-1) \tag{10}$$

Algorithm 2 Activation Calculation

Input: Q, K
1: **for** i = 0 to (11 × N - 1) **do**
2: **for** h = 0 to (H - 1) **do**
3: $A[h \times H + h] += q[h][i] \times k[i][h]$
4: **for** m = 0 to (h - 1) **do**
5: $A[h \times H + m] += q[h][i] \times k[i][m]$
6: $A[m \times H + h] += q[m][i] \times k[i][h]$
7: **end for**
8: **end for**
9: **end for**
10: **for** i = 0 to (11 × N - 1) **do**
11: $\tilde{A}[i] = \sigma(A[i])$
12: **end for**
Output: $\tilde{A} = \sigma(A)$

Algorithm 3 Output Calculation

Input: $\tilde{A}$, V
1: **for** i = 0 to (H - 1) **do**
2: **for** j = 0 to (H - 1) **do**
3: **for** k = 0 to (11 × N - 1) **do**
4: $Attn\_out[k] = \tilde{A}\ [i \times H + j]$
$\times V[k + 11 \times N \times j]\ \times W_o$
5: **end for**
6: **end for**
7: **end for**
Output: *Attn_out* matrix

where $a_{i,m}$ is an element of the result of KQ^T, and H corresponds to the number of output channels of a feature matrix (Q, K, V). Based on the structure of data flow transmission, an optimized matrix calculation process is designed. When the $q_{i,j}$ and $k_{i,j}$ are read from the Q, K FIFO queues, $k_{i,j}q_{j,i}$ to $a_{i,i}$ can be added because $q_{i,j}$ are $q_{j,i}$ in Q^T. And as shown in (11), the $q_{j,i}$ and $k_{i,j}$ are also used to get the product with the cached K and Q elements, respectively. The result is added to the elements at the corresponding positions of the KQ^T matrix. Due to the matrix of Q, K being dynamically generated, the calculation process is not static, and we call it the dynamic matrix multiplication in FPGA

$$\begin{aligned} a_{z,i} &+= k_{z,j}q_{j,i}, z \in (0, 1, \ldots, i-1) \\ a_{i,w} &+= k_{i,j}q_{j,w}, w \in (0, 1, \ldots, i-1). \end{aligned} \tag{11}$$

The essence of (11) is to split (10) to make it suitable for the data flow queue of Q and K. Since the values of the Q and K matrices are dynamically generated, the activation results ($\tilde{A}$) in (4) can not be computed unless obtaining all data in the Q and K FIFO queues. Algorithm 2 shows the whole process of this module.

Output Calculation Module Design: This module performs the matrix multiplication of $\tilde{A}$ with V and also conducts the final convolution layer. In this process, the matrix multiplication is much easier than that of calculating the dynamic matrix due to the matrix V is known.

As shown in Algorithm 3, an element $\tilde{a}_{i,j}$ of the current input $\tilde{A}$ is multiplied by the element v_j of the jth row of V, and then the result is accumulated to the element r_i of the ith row of the result matrix. The calculation process is shown in

$$r_{i,m} += \tilde{a}_{i,j}v_{j,m},\ \ m \in (0, 1, \ldots, 11 \times N - 1). \tag{12}$$

Since the value of the V matrix is completely computed, when the $\tilde{A}$ matrix inputs a row of data, a row of data of $\tilde{A}V$ can be calculated as

$$s_{o,i} = W_o \sum_{j=0}^{H-1} r_{i,j} \tag{13}$$

where W_o represents the weight value corresponding to the oth output channel of the output convolution function, and $s_{o,i}$ is the ith output value corresponding oth output channel. In this article, the number of output channels is set to one so that the input and output series of the self-attention layer have the same length.

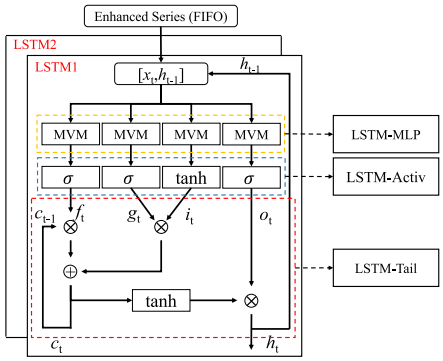


Fig. 8. Step computation of LSTM cell.

2) MD-LSTM Pipeline Design: Following the same design principles as that of the SAM layer design, the design of the MD-LSTM cell uses the FIFO queue and the data flow to transmit input values, intermediate results serially, and output values. So the pipeline of the entire cell calculation process is realized. By separating the depth dimension and the temporal dimension of the MD-LSTM cell, these two dimensions' data flow is entirely run in parallel. Thus, a specified LSTM cell, which can apply to the calculation process of the two dimensions, needs to be carefully designed.

First, the calculation process of a single time step of the LSTM network is analyzed. Since the weights matrix and bias vectors of the fully connected layer inside the LSTM cell are completely known, to minimize the time delay, the whole computation does not need to wait for the fully connected layer to complete its calculation. So, after calculating one row of the output result, the following calculation step can be started immediately.

As shown in Fig. 8, the calculation process of the LSTM cell is restructured into three parts, including LSTM-MLP, LSTM-Activ, and LSTM-Tail. The three modules are connected through FIFO resources to realize the task-level pipeline. Various optimization methods of HLS are appropriately used inside each module to reduce computing delay and improve throughput.

LSTM-MLP: The LSTM-MLP module is used to process Matrix-Vector Multiplication in parallel. In the internal calculation of LSTM, four gate signals are independent, so the optimization method of loop unrolling in HLS is used to the for loop of line 3 in Algorithm 4 to perform the four MVM parallelly.

LSTM-Activ: The LSTM-Activ module is not time-dependent so the activation value can be quickly calculated

Algorithm 4 LSTM-MLP Calculation

Input: *Attn_out* matrix
1: **for** t = 0 to (N - 1) **do**
2: **for** j = 0 to 10 **do**
3: **for** k = 0 to (4 × H - 1) **do**
4: $gifo_t[k] += W_x[k] \times Attn\_out[j + j \times N] + W_h \times h_{(t-1)}[j]$
5: **end for**
6: **end for**
7: **for** j = 11 to (H - 1) **do**
8: **for** k = 0 to (4 × H - 1) **do**
9: $gifo_t[k] += W_h \times h_{(t-1)}[j]$
10: **end for**
11: **end for**
12: **end for**
Output: *gifo* matrix

Algorithm 5 LSTM-Activ

Input: *gifo* matrix
1: **for** t = 0 to (N - 1) **do**
2: **for** k = 0 to (H - 1) **do**
3: $g_t[k] = \tanh(gifo_t[k])$
4: $i_t[k] = \sigma(gifo_t[k + H])$
5: $f_t[k] = \sigma(gifo_t[k + 2 \times H])$
6: $o_t[k] = \sigma(gifo_t[k + 3 \times H])$
7: **end for**
8: **end for**
Output: g_t, i_t, f_t, o_t

Algorithm 6 LSTM-Tail

Input: i_t, f_t, g_t, o_t
1: **for** t = 0 to (N -1) **do**
2: **for** i = 0 to (H - 1) **do**
3: $c_t[i] = g_t[i] \times i_t[i] + c_{t-1}[i] \times f_t[i]$
4: $h_t[i] = \tanh(c_t[i]) \times o_t[i]$
5: **end for**
6: **end for**
Output: c_t, h_t

for the next module. By using lookup table optimization, two activation functions are implemented in the LSTM-Active module, including the sigmoid and tanh functions. Algorithm 5 shows the whole process of this module.

LSTM-Tail: The LSTM-Tail module is applied to calculate the output value of both the final hidden layer unit and the memory unit. Since the calculation of the hidden unit depends on that of the memory unit, these two steps of lines 3 and 4 in Algorithm 6 cannot be parallelized.

By limiting the module's interface as a FIFO queue, we only need to focus on the parallel optimization within the module in our FPGA-based pipeline design scheme. So, the FPGA-based model design can be efficiently implemented.

VII. Experiments and Evaluation

We have designed and trained different baseline comparison models. The generated dataset includes a total of 2813144 traces across six types of attacks. Before training the model, we manually partition the dataset, allocating approximately 80% of the data–about 2250731 traces–for training. A further 10%–roughly 281314 entries–was used for testing, with the remaining 10% reserved for validation. Then the computation architecture of MULSAM was redesigned to be suitable for deployment on the Ultra96-V2 board. The PC (Intel i9-10850K CPU @ 3.6 GHz × 10, NVIDIA Quadro RTX 4000 GPU @ 8 GB) runs Windows 10, while the Ultra96-V2 (2-GB LPDDR4, UltraScale+MPSoC) runs PYNQ.

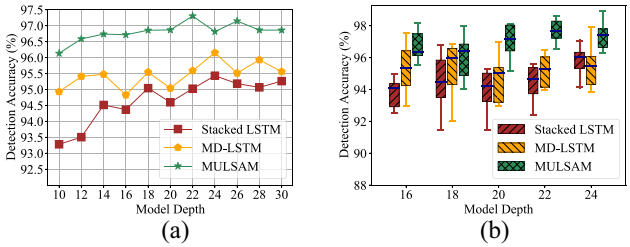


Fig. 9. Performance evaluation on different model depth. (a) Overall accuracy. (b) Training stability.

Our evaluation focuses on four aspects of the performance: different model depths; various ML models; and two above mentioned models based on the FPGA platform with different datasets.

A. Different Model Depths

In this section, the normal and attack datasets are used, which contain the original data from the vehicular CAN bus and the generated attack data, respectively, to evaluate the performance of Stacked LSTM, MD-LSTM, and our MULSAM with different model depths. To compare the performance improvement of the SAM on the MULSAM, the structure of the self-attention layer is fixed, and only the model depth of the MD-LSTM component was changed. The range of model depth is 10-30, and the step size is 2.

In our experimental configuration, the Stacked LSTM model employs two layers of LSTM, the MD-LSTM comprises four LSTM layers, and the MULSAM integrates four LSTM layers as well. An increased number of LSTM layers can potentially enhance detection accuracy, yet escalate computational resource overhead. Our experiments demonstrate that a four-layer LSTM already achieves a high-detection accuracy, sufficient for practical needs while at the same time without incurring redundant resource overhead. Further increasing LSTM layers yields negligible improvements in detection accuracy while incurring extra computational resources.

Overall Accuracy: Three models, including MD-LSTM, Stacked LSTM, and MULSAM, are used to test the detection accuracy. The detection accuracy present is the average value from multiple experiments to prevent the dropout layer from affecting the stability of the results. As shown in Fig. 9(a), the detection accuracy of MULSAM is 1-2% higher than the other two models under all depth conditions, and the accuracy of all models tends to increase with the increment of the models' depth. However, there exists a peak with all models, which means that the accuracy of models no longer improves when they reach a certain depth. MULSAM peaks can be seen at a model depth of 22 and 24 for the other two models.

Training Stability: The stability of the training process is crucial to obtaining a robust model. Fig. 9(b) visualizes the accuracy of different methods when the number of model depths is 16, 18, 20, 22, and 24. The MULSAM has an accuracy fluctuation range of about 2-4% at all model depths, while Stacked LSTM has a range of about 2.4-6%, and MD-LSTM has a range of about 2.4-5%. A smaller range means that it's possible to faster train a model with high accuracy.

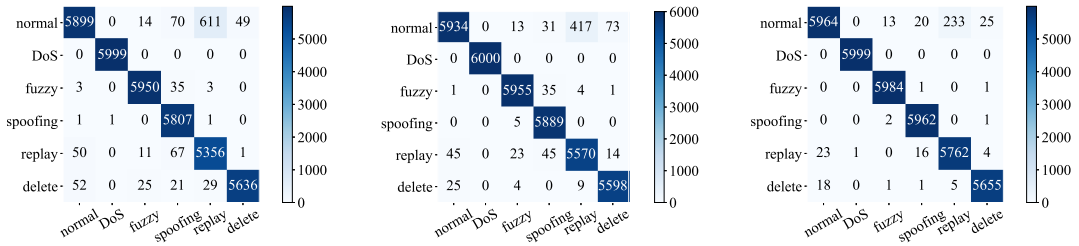


Fig. 10. Confusion matrix of classification. (a) Stacked LSTM. (b) MD-LSTM. (c) MULSAM.

TABLE I
MODEL INITIAL HYPERPARAMETERS

Parameters	Value
Epochs	100
Early stopping	5
Activation Function	Softmax
Learning rate	1e-3
optimizer	Adam
Loss Function	CrossEntropyLoss
Batch size	128
Steps	32

TABLE II
PERFORMANCE ON VARIOUS MODELS

Model	Acc (%)	Attack	Recall	P	F_1
SVM	82.07	normal	0.2623	0.1456	0.1008
		DoS	0.9983	0.9953	0.9923
		fuzzy	0.9439	0.9650	0.9870
		spoofing	0.9814	0.9272	0.8787
		replay	0.7428	0.7023	0.6660
		delete	0.5968	0.6909	0.8202
MLP	80.08	normal	0.5410	0.7795	0.6387
		DoS	0.9988	0.9463	0.9718
		fuzzy	0.9754	0.9718	0.9736
		spoofing	0.8814	0.9540	0.9163
		replay	0.8400	0.6902	0.7577
		delete	0.6396	0.4444	0.5244
CNN	87.64	normal	0.6711	0.8083	0.7333
		DoS	0.9980	0.9773	0.9875
		fuzzy	0.9525	0.9687	0.9605
		spoofing	0.9956	0.9787	0.9871
		replay	0.8778	0.7312	0.7978
		delete	0.8044	0.7900	0.7972
Stacked LSTM	97.07	normal	0.8880	0.9823	0.9328
		DoS	**1.0000**	0.9998	0.9999
		fuzzy	0.9932	0.9917	0.9924
		spoofing	0.9995	0.9678	0.9834
		replay	0.9765	0.8927	0.9327
		delete	0.9780	0.9912	0.9845
MD-LSTM	97.91	normal	0.9174	0.9882	0.9515
		DoS	**1.0000**	**1.0000**	**1.0000**
		fuzzy	0.9932	0.9925	0.9928
		spoofing	0.9992	0.9815	0.9902
		replay	0.9777	0.9283	0.9524
		delete	0.9933	0.9845	0.9889
MULSAM	**98.98**	normal	**0.9535**	**0.9932**	**0.9729**
		DoS	**1.0000**	0.9998	0.9999
		fuzzy	**0.9997**	**0.9973**	**0.9985**
		spoofing	**0.9995**	**0.9937**	**0.9966**
		replay	**0.9924**	**0.9605**	**0.9762**
		delete	**0.9956**	**0.9945**	**0.9951**

Differences in Classification: As shown in Fig. 10, the corresponding confusion matrices are generated to analyze the classification differences of the models. In the classification results of the replay attack, which is the most malicious among all attacks, all three models identify the attack as a normal state. However, the number of misidentifications by MULSAM is only 233, which is approximately half that of MD-LSTM and one-third for Stacked LSTM. The detection difference in the replay attack may be caused by the multidimensional architecture and the self-attention layer in MULSAM. This demonstrates that MULSAM will have a lower-false-positive rate and enhance the characteristics of small-batch attacks.

B. Various Machine Learning Models

Our baseline comparison models include SVM, MLP, CNN, stacked LSTM, and MD-LSTM. The evaluation metrics of comparison include the overall accuracy, precision, recall, and F_1 score. The details of the metrics are as follows: Accuracy $= ([\mathrm{TP}+\mathrm{TN}]/[\mathrm{TP}+\mathrm{FN}+\mathrm{FP}+\mathrm{TN}])$, Precision $= (\mathrm{TP}/[\mathrm{TP}+\mathrm{FP}])$, Recall $= (\mathrm{TP}/[\mathrm{TP}+\mathrm{FN}])$, and $F_1 = (2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}/[\mathrm{Precision}+\mathrm{Recall}])$, where TP, FP, TN, and FN are four outcomes of the classification, representing true positive, false positive, true negative, and false negative, respectively.

The cross-entropy loss function, as defined as $H(p,q) = -\sum_x p(x)\log q(x)$, is used as the loss function for all ML models, where p is the expected result, q stands for the predicted result, and x is the index for both. All the models are trained on 80 percent of the CAN data and validated on 10 percent, while the remaining 10 percent is used as a testing set. As Table I shows, the maximum number of iterations is set to 100, and the initial learning rate is 1e-3. To reduce redundant training process, the training of each model can automatically be terminated early by setting the stopping threshold, which is a hyperparameter debugged and selected according to the validation set. The adaptive gradient algorithm (Adam) [27], which can adjust the learning rate, is used to optimize our models. To prevent over-fitting of these models, the dropout technique [28] is used in all deep learning models. The total steps of the input series are set to 32, which corresponds to approximately 30ms of CAN messages.

As we can see the performance of various models in Table II, the SVM, CNN, and MLP perform well in detecting attacks with flooding properties but perform poorly on normal state and attacks with small-batch, which causes the overall accuracy to be less than 90%, while that of LSTM-related models is more than 97%. For example, the recall of SVM, MLP, and CNN are only 26.25%, 54.10%, and 67.11%, respectively. The superior performance of LSTM-related models shows the structure of LSTM is suitable for processing the data with time correlation, while the forms of SVM, MLP, CNN fail to do so. MULSAM has a recall of 95.35% on the detection of the normal state, while

TABLE III
SELECTED DEPTH AND CORRESPONDING ACCURACY

Model	Depth	Acc (%)
Stacked LSTM	24	97.07
MD-LSTM	24	97.91
MULSAM	22	98.98

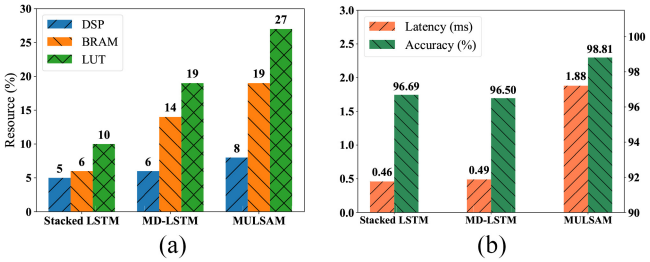


Fig. 11. Performance evaluation on FPGA device. (a) Resource overhead. (b) Accuracy and latency.

Stacked LSTM is only 88.80% and MD-LSTM is 91.74%, which means that MULSAM can provide a more credible basis for vehicle to make a decision. In the detection of the spoofing attack with the flooding feature, MULSAM has a precision of 99.37%, while Stacked LSTM is only 96.78% and MD-LSTM is 98.15%. The performance of the three LSTM-related models is close, and all exceed 99% in the detection of DoS and fuzzy attacks. In addition to having a slight drop in the results of the DoS attack, the MULSAM has a higher recall, precision, and F_1 compared with other models.

C. Hardware Acceleration Evaluations

The results of the implementations of three LSTM-related models, including stacked LSTM, MD-LSTM, and MULSAM, are presented in this section. Due to the advantages of the modular design approach, the programming of the LSTM cell in MULSAM can be conveniently reused by Stacked LSTM and MD-LSTM. The evaluations on FPGA platform and the comparison between FPGA and TX2 will be presented.

1) Performance of FPGA Platform: The embedded experiments for Stacked LSTM, MD-LSTM, and MULSAM are running on an Ultra96-V2 device with 70560 look up table (LUT), 216 block random access memory (BRAM) and 360 digital signal processor (DSP). The model depth of Stacked LSTM, MD-LSTM, and MULSAM are selected corresponding to the highest accuracy in Fig. 9(a) for the accelerated model design on the FPGA platform. The depth selection and the corresponding accuracy of the three test models are shown in Table III.

Resource Overhead: The resource overhead has been visualized after the model has been synthesized to register transfer level (RTL) code, as shown in Fig. 11(a). It can be seen from Fig. 11(a) that MULSAM uses the most resources because of its complex internal structure. A MD-LSTM cell contains 2-D LSTM, so the resource overhead of MD-LSTM is approximately twice that of Stacked LSTM of the same model depth. As we can see, all three accelerators are very resource-efficient, which helps reduce power consumption in FPGA.

Accuracy and Latency: The experiment data includes 1600 evenly distributed samples, randomly extracted from the test datasets and the corresponding labels. Fig. 11(b) shows the latency and the accuracy of different FPGA-based accelerators. Compared with the accuracy of the full-precision models in Table III, the fixed-point quantization in the FPGA-based accelerators results in that of Stacked LSTM, MD-LSTM, and MULSAM in 0.38%, 1.41%, and 0.17% reduction, respectively. MD-LSTM has the largest loss of accuracy, which is likely that the multidimensional architecture of MD-LSTM causes its performance to be more sensitive to the weight parameters. However, MULSAM also has a multidimensional architecture. Still, it has the lowest-drop accuracy (98.81%), which is because that the SAM reduces the dependency of MULSAM on the weight parameters. It is worth noting that the total time step of the test data in the experiment is 32 steps, which corresponds to about 30 ms of CAN data on the CAN bus network. Therefore, the latency performance of all models (maximum 1.88 ms) is far less than the generated time of CAN data, which positively impacts the deployment of embedded FPGA devices.

2) Comparison Between FPGA and TX2: To compare FPGA-based embedded system security with GPU-based embedded platform, we designed and trained different baseline comparison models on Jetson TX2 as the Commercial-Off-The-Shelf GPU-based embedded platform. The Jetson TX2 is an embedded computing device equipped with 56-core NVIDIA Pascal GPU architecture with 256 NVIDIA CUDA cores and dual-core NVIDIA Denver2 + quad-core ARM Cortex-A57. Both Ultra96-V2 and TX2 run PYNQ.

Throughput Rate: Fig. 12(a) shows the throughput rate of different LSTM-related accelerators in different platforms. The same models running on different platforms have the same network topology and weight parameters. We can see that the throughput rate of the FPGA platform is larger than that of the TX2 platform due to its powerful computing performance. In the throughput rates of the FPGA platform, MD-LSTM has that of 1316.56 MFLOP/s because the two independent dimensional LSTM layers inside the MD-LSTM utilize a parallel design methodology.

Power and Energy Efficiency: From Fig. 12(b), the power consumption of all three models exceeds 5 W on the TX2, while that on the FPGA platform is only about 2 W. In terms of energy efficiency, the FPGA-based implementation still has a great advantage, which can be seen in Fig. 12(c). For Stacked LSTM models, the FPGA hardware accelerators are about twenty-one times as energy-efficient as the TX2. The energy efficiency of MULSAM is 145 MLOP/(s·W), which is much higher than the 11 MLOP/(s·W) of the TX2 platform. Although MULSAM has the lowest-energy efficiency among the three FPGA-based accelerators, its energy efficiency is still higher than TX2 which is the mainstream GPU platform for embedded computing.

D. Models on the Move

We conducted testing experiments on an Ultra96-V2 embedded device installed in an actual vehicle as an FPGA-enabled

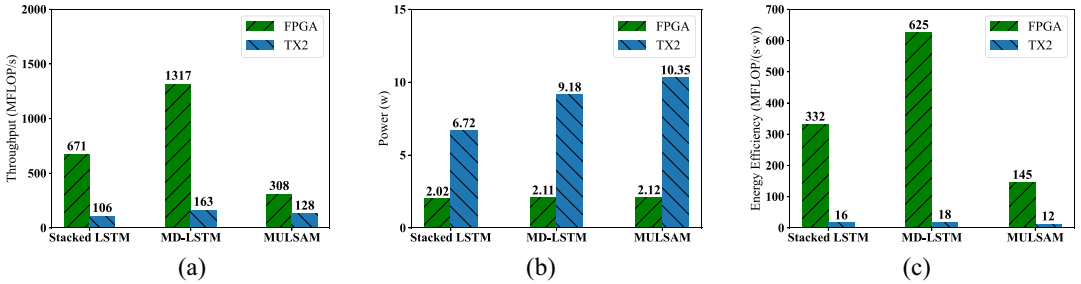


Fig. 12. Comparison of different accelerators between FPGA and TX2 platforms. (a) Throughput rate. (b) Power. (c) Energy efficiency.

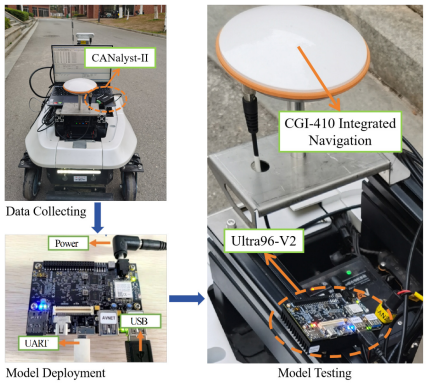


Fig. 13. FPGA device during testing.

TABLE IV
PERFORMANCE ON VARIOUS MODELS

Model	Acc (%)	Attack	Recall	P	F_1	FPR	FNR
Stacked LSTM	95.25	normal	1.0000	0.9820	0.9909	0.0037	0.0000
		DoS	1.0000	0.9554	0.9772	0.0090	0.0000
		fuzzy	1.0000	1.0000	1.0000	0.0000	0.0000
		spoofing	0.9463	0.7856	0.8585	0.0424	0.0537
		replay	0.9751	0.9989	0.9869	0.0002	0.0249
		delete	0.8037	0.9928	0.8883	0.0013	0.1963
MD-LSTM	89.90	normal	0.9513	0.9936	0.9720	0.0013	0.0487
		DoS	1.0000	1.0000	1.0000	0.0000	0.0000
		fuzzy	1.0000	1.0000	1.0000	0.0000	0.0000
		spoofing	1.0000	0.7675	0.8685	0.0454	0.0000
		replay	0.7366	0.9352	0.8241	0.0140	0.2634
		delete	0.7685	0.6975	0.7313	0.0526	0.2315
MULSAM	**97.47**	normal	0.9968	**1.0000**	**0.9984**	**0.0000**	0.0032
		DoS	**1.0000**	**1.0000**	**1.0000**	**0.0000**	**0.0000**
		fuzzy	**1.0000**	**1.0000**	**1.0000**	**0.0000**	**0.0000**
		spoofing	0.9977	**0.9140**	**0.9540**	**0.0173**	0.0023
		replay	0.9187	0.9352	0.9269	0.0133	0.0813
		delete	**0.9327**	**0.9988**	**0.9646**	**0.0002**	0.0673

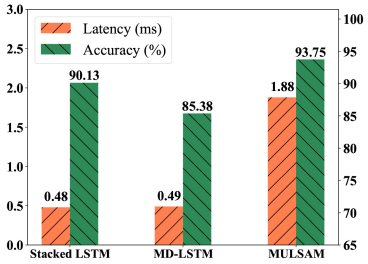


Fig. 14. FPGA performance using data from autonomous vehicle platform.

gateway for intrusion detection on vehicular CAN bus, as shown in Fig. 13. The embedded experiments for Stacked LSTM, MD-LSTM, and MULSAM are run based on their trained models from the datasets collected in real world and tested in the driving scenario to guarantee that the vehicle can prevent malicious navigation in a short time from corresponding attacks.

Our experiment utilizes an autonomous vehicle platform measuring 1000 mm × 705 mm × 389 mm. The track width between the left and right wheels is 487 mm, while the wheelbase–the distance between the front and rear axles is 610 mm. The diameter of the drive wheels is 241 mm. The autonomous vehicle platform is controlled by an STM32 industrial computer equipped with a speed of 1 Mb/s CAN communication interface, which manages the two drive wheels to enable linear movement or differential speed for curved motion. We employ a USB-CAN analyzer (CANalyst-II) and a laptop installed with USB-CAN Tool software to collect CAN data from the autonomous vehicle platform. In our experiment, we designed corresponding attack effects during the operation of the autonomous vehicle platform to simulate the actual driving situation of the real vehicle in the actual environment.

We trained and evaluated three models using a total of 151050 pieces of data collected from the autonomous vehicle platform. The training set, test set, and validation set were divided into an 8:1:1 ratio. The results when run on a PC are shown in Table IV. Test results after deploying the model to the FPGA platform as shown in Fig. 14, the test data includes 1600 evenly distributed samples, randomly extracted from the test datasets and the corresponding labels.

The results in Table IV and Fig. 14 demonstrate that the detection accuracy of MULSAM is significantly superior to that of Stacked LSTM and MD-LSTM, regardless of the running platform. Compared to models trained with open-source datasets, the models trained with our self-collected data show a slight decrease in detection accuracy. This is due to the smaller data volume of our self-collected data. Our collected dataset is only about 150 000, while the open-source dataset contains more than 2.8 million traces. However, MULSAM's detection accuracy still exceeds 93% with a much smaller dataset, which reflects a low dependency on the dataset and a strong adaptability to different datasets. We note that although MULSAM costs an additional 1.4 ms in computational overhead compared to the other two models, tolerating a small amount of time delay to achieve a 2% improvement in accuracy is entirely acceptable and meaningful, considering that the fastest human reaction time is around

TABLE V
PERFORMANCE ON VARIOUS DATASETS

Dataset	Model	Acc (%)	Attack	Recall	P	F_1	FPR	FNR
Sonata	LSTM	99.69	benign	0.9952	0.9984	0.9968	0.0012	0.0048
			flooding	1.0000	0.9952	0.9976	0.0013	0.0000
			fuzzy	0.9961	0.9942	0.9952	0.0013	0.0039
			malfunction	1.0000	1.0000	1.0000	0.0000	0.0000
	MULSAM	99.67	benign	0.9936	**1.0000**	0.9968	**0.0000**	0.0064
			flooding	1.0000	0.9952	0.9976	0.0013	0.0000
			fuzzy	**1.0000**	0.9904	0.9952	0.0021	**0.0000**
			malfunction	1.0000	1.0000	1.0000	0.0000	0.0000
Soul	LSTM	99.16	benign	0.9880	0.9945	0.9913	0.0042	0.0120
			flooding	1.0000	0.9979	0.9989	0.0006	0.0000
			fuzzy	0.9922	0.9759	0.9840	0.0067	0.0078
			malfunction	0.9887	0.9981	0.9934	0.0003	0.0113
	MULSAM	**99.71**	benign	**0.9919**	**0.9995**	**0.9956**	**0.0004**	**0.0081**
			flooding	0.9989	0.9968	0.9979	0.0009	0.0011
			fuzzy	**1.0000**	**0.9868**	**0.9934**	**0.0036**	**0.0000**
			malfunction	**1.0000**	**1.0000**	**1.0000**	**0.0000**	**0.0000**
Spark	LSTM	98.38	benign	0.9945	0.9742	0.9843	0.0257	0.0055
			flooding	1.0000	1.0000	1.0000	0.0000	0.0000
			fuzzy	0.8452	0.9632	0.9003	0.0030	0.1548
			malfunction	0.9967	0.9967	0.9967	0.0006	0.0033
	MULSAM	**98.92**	benign	0.9924	**0.9871**	**0.9898**	**0.0130**	0.0076
			flooding	1.0000	0.9979	0.9990	0.0007	0.0000
			fuzzy	**0.9085**	0.9485	**0.9281**	0.0041	**0.0915**
			malfunction	**1.0000**	**1.0000**	**1.0000**	**0.0000**	**0.0000**

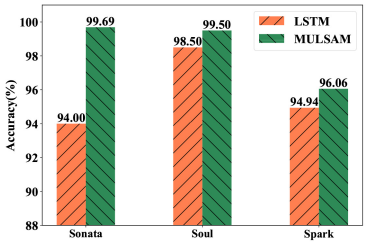


Fig. 15. FPGA performance evaluation using various datasets.

0.7s and autonomous vehicles may require 0.5s in driving scenarios.

E. Different Dataset Comparison

We compared the performance of the MULSAM algorithm with the algorithm used in [13], which employed a single-layer LSTM for attack detection and utilized three different datasets: 1) the Sonata dataset; 2) the Soul dataset; and 3) the Spark dataset, to train their algorithm. We trained both a single-layer LSTM and MULSAM using the datasets from the [13] and compared the detection results. The results when run on a PC are as shown in Table V.

The results in Table V show that the detection accuracy, Recall, and F1 score of MULSAM are slightly higher than those of single-layer LSTM across the Sonata, Soul, and Spark datasets. In addition, MULSAM shows lower FNR and FPR than single-layer LSTM spanning across all three datasets. Especially in the Spark dataset, our MULSAM has a 0.29% decrease in FPR and a 1.61% decrease in FNR, further indicating that MULSAM has a strong adaptability to various datasets and is more likely to exhibit better performance in complex real-world application scenarios.

We also employ our method on the FPGA platform using the dataset in [13]. The results are shown in Fig. 15, the test data also includes 1600 evenly distributed samples, randomly extracted from the test datasets.

Fig. 15 demonstrates that the superiority of the MULSAM can be further improved after deploying our MULSAM to the FPGA platform. The average detection accuracy of MULSAM is 2.6% higher than that of single-layer LSTM. Furthermore, the detection accuracy of MULSAM deployed on the FPGA platform degrades by only 2.8%, while the performance of single-layer LSTM reduces by 5.6%, which reflects a higher reliability of MULSAM than single-layer LSTM.

VIII. DISCUSSION

Power Consumption: Compared to typical ECUs, implementing our method on FPGA does require higher-power consumption. However, our experiment result shows that this power consumption only costs 1.7% of the total power consumption of an autonomous vehicle platform, which is mounted with a 51.2 V/40 Ah Li-ion battery pack and can support 12 V and 120 W power supply. We note that it is acceptable to sacrifice a small part of the power consumption to ensure high-recognition accuracy, especially in driving scenarios where safety is the highest priority.

One-Shot Attack: In our MULSAM algorithm, we integrate a SAM with MD-LSTM to enhance the algorithm's ability to learn and process time-series data, thereby improving the detection accuracy of single-time attacks within vehicular networks, and effectively addressing the problem of single-time attacks. The SAM can consider the other elements within the sequence when processing a certain element, capturing the long-distance dependencies within the sequence, thus posing a powerful capability to detect anomalies in sequences. Meanwhile, MD-LSTM enhances the original LSTM model by introducing a multidimension mechanism, which enables long-term storage and short-term memory of sequence information while processing information at different time dimensions, allowing it to perform distinctly in dealing with anomalies in data sequences. This combination within the MULSAM algorithm leverages the advantages of both the SAM and MD-LSTM in detecting anomalies in time-series data, such as single-time attacks, enabling identification and response to those potentially one-off but destructive attack behaviors, thereby securing the safety of vehicular networks.

DoS and Fuzzy Attacks With Nonflooding: Flooding strategy is a common approach in DoS and fuzzy attacks. However, a few DoS and fuzzy attacks are based on nonflooding strategies, such as sending specific constructed CAN frames and sending abnormal or invalid CAN messages, to interfere with the normal communication of the network. However, those DoS and fuzzy attacks are still based on detectable patterns with unique characteristics. As a result, we can train a new model to enable the detection of both DoS and fuzzy attacks with nonflooding characteristics.

Escape Detection: The powerful sequence anomaly detection capabilities in our proposed MULSAM algorithm can effectively identify most replay and delete attacks. However, these two types of attack are slightly shifted from the normal CAN signals, which may not be fully distinguishable by our MULSAM. For the few attacks that escape detection, we can further enhance the system's detection capabilities

by combining multiple detection methods or by setting up multiple layers of defense mechanisms.

Computational Overhead: The proposed MULSTM algorithm aims to enhance the accuracy of attack detection. Given that the fastest human reaction time can be around 0.7 s and an autonomous vehicle may require 0.5 s, although MULSTM requires more computational resources and costs an additional 1.4 ms in computation time compared to the other two algorithms, sacrificing a small portion of computational resources and a slight delay while in exchange for a 2% improvement in accuracy is entirely acceptable and meaningful in practice. In an era where the computational ability of autonomous driving devices is boosting, tolerating a small portion of computing resources to ensure high accuracy is a strategy worth exploring and implementing.

IX. Conclusion

The security threats along with the development of autonomous driving affect the benign message transmission of the in-vehicle CAN bus communication network. We developed an enhanced intrusion detection technology based on MD-LSTM and SAM (MULSAM), which can detect the type of attack with a small-batch. To deploy our model on vehicles, the computation process of MULSAM adopted multiple parallel methods and implemented them based on FPGA platform. The experiment proved that the lightweight reduction of the weights of the neural network did not impact the detection accuracy, providing a promising solution for vehicular real-time online detection systems.

References

[1] K. H. Johansson, M. Törngren, and L. Nielsen, "Vehicle applications of controller area network," in *Handbook of Networked and Embedded Control Systems*. Berlin, Germany: Springer, 2005, pp. 741–765.

[2] O. Avatefipour and H. Malik, "State-of-the-art survey on in-vehicle network communication (CAN-Bus) security and vulnerabilities," 2018, *arXiv:1802.01725*.

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

[4] H. J. Jo, J. H. Kim, H.-Y. Choi, W. Choi, D. H. Lee, and I. Lee, "MAuth-CAN: Masquerade-attack-proof authentication for in-vehicle networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2204–2218, Feb. 2020.

[5] S. Rajapaksha, G. Madzudzo, H. Kalutarage, A. Petrovski, and M. O. Al-Kadri, "CAN-MIRGU: A comprehensive CAN bus attack dataset from moving vehicles for intrusion detection system evaluation," in *Proc. Symp. Veh. Security Privacy Internet Soc.*, 2024, pp. 1–11.

[6] E. Aliwa, O. Rana, C. Perera, and P. Burnap, "Cyberattacks and countermeasures for in-vehicle networks," *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–37, 2021.

[7] Z. Tan et al., "Human–machine interaction in intelligent and connected vehicles: A review of status quo, issues, and opportunities," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 13954–13975, Sep. 2022.

[8] M. Baek et al., "Accurate prediction of protein structures and interactions using a three-track neural network," *Science*, vol. 373, no. 6557, pp. 871–876, 2021.

[9] D. Wu, H. Xu, Z. Jiang, W. Yu, X. Wei, and J. Lu, "EdgeLSTM: Towards deep and sequential edge computing for IoT applications," *IEEE/ACM Trans. Netw.*, vol. 29, no. 4, pp. 1895–1908, Aug. 2021.

[10] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[11] K. Kim, J. S. Kim, S. Jeong, J.-H. Park, and H. K. Kim, "Cybersecurity for autonomous vehicles: Review of attacks and defense," *Comput. Security*, vol. 103, Apr. 2021, Art. no. 102150.

[12] T. Huang, J. Zhou, and A. Bytes, "ATG: An attack traffic generation tool for security testing of in-vehicle CAN bus," in *Proc. 13th Int. Conf. Availab., Rel. Security*, 2018, pp. 1–6.

[13] V. K. Kukkala, S. V. Thiruloga, and S. Pasricha, "INDRA: Intrusion detection using recurrent autoencoders in automotive embedded systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3698–3710, Nov. 2020.

[14] H. Xu, D. Wu, Y. Lu, J. Lu, and H. Zeng, "Models on the move: Towards feasible embedded AI for intrusion detection on vehicular CAN bus," in *Proc. USENIX ATC*, 2024, pp. 1049–1063.

[15] O. Y. Al-Jarrah, C. Maple, M. Dianati, D. Oxtoby, and A. Mouzakitis, "Intrusion detection systems for intra-vehicle networks: A review," *IEEE Access*, vol. 7, pp. 21266–21289, 2019.

[16] I. Rouf et al., "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study," in *Proc. USENIX Security Symp.*, 2010, pp. 323–338.

[17] 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.

[18] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "LSTM-based intrusion detection system for in-vehicle can bus communications," *IEEE Access*, vol. 8, pp. 185489–185502, 2020.

[19] T. Hoppe, S. Kiltz, and J. Dittmann, "Applying intrusion detection to automotive it-early insights and remaining challenges," *J. Inf. Assur. Security*, vol. 4, no. 6, pp. 226–235, 2009.

[20] T. P. Vuong, G. Loukas, and D. Gan, "Performance evaluation of cyber–physical intrusion detection on a robotic vehicle," in *Proc. CIT/IUCC/DASC/PICom*, 2015, pp. 2106–2113.

[21] C. Ling and D. Feng, "An algorithm for detection of malicious messages on CAN buses," in *Proc. Nat. Conf. Inf. Technol. Comput. Sci.*, 2012, pp. 627–630.

[22] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *Proc. 25th USENIX Conf. Security Symp.*, 2016, pp. 911–927.

[23] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "VoltageIDS: Low-level communication characteristics for automotive intrusion detection system," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 2114–2129, 2018.

[24] J. Zhou, P. Joshi, H. Zeng, and R. Li, "BTMonitor: Bit-time-based intrusion detection and attacker identification in controller area network," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 6, pp. 1–23, 2019.

[25] G. Dupont, A. Lekidis, J. J. den Hartog, and S. S. Etalle, 2019, "Automotive controller area network (CAN) bus intrusion dataset v2," Dataset. [Online]. Available: https://data.4tu.nl/articles/dataset/Automotive_Controller_Area_Network_CAN_Bus_Intrusion_Dataset/12696950/2

[26] M. Siracusa and F. Ferrandi, "Tensor optimization for high-level synthesis design flows," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 4217–4228, Nov. 2020.

[27] P. Joulani, A. Gyorgy, and C. Szepesvári, "Delay-tolerant online convex optimization: Unified analysis and adaptive-gradient algorithms," in *Proc. AAAI*, 2016, pp. 1744–1750.

[28] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *Proc. 30th Conf. Neural Inf. Process. Syst.*, 2016, pp. 1163–1171.

He Xu received the B.S. degree from the College of Electrical and Information Engineering, Hunan University, Changsha, China, in 2019, where he is currently pursuing the M.S. degree in electronic science and technology.

His research interests include deep learning and embedded systems.

Xiaokang Shi received the B.S. degree in electronic information engineering from Huaqiao University, Xiamen, China, in 2022. He is currently pursuing the M.S. degree with the College of Electrical and Information Engineering, Hunan University, Changsha, China.

His research interests include wireless sensing and embedded systems.

Hansheng Liu received the B.S. degree in mechanical design manufacturing and automation from Fujian Agriculture and Forestry University, Fuzhou, China, in 2021. He is currently pursuing the M.S. degree with the College of Mechanical and Vehicle Engineering, Hunan University, Changsha, China.

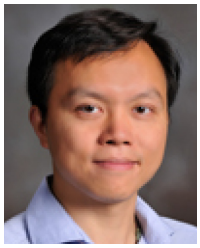
His research interests include simultaneous localization and mapping.

Yanwen Wang (Member, IEEE) received the M.S. degree in electrical engineering from the Missouri University of Science and Technology, Rolla, MO, USA, in 2013, and the Ph.D. degree in computer science from Hong Kong Polytechnic University, Hong Kong, in 2019.

He is currently an Associate Professor with the College of Electrical and Information Engineering, Hunan University, Changsha, China. His research interests include mobile computing, signal processing, and data analysis.

Jiwu Lu (Member, IEEE) received the M.S. degree from Siegen University, Siegen, German, in 2006, and the Ph.D. degree from Twente University, Enschede, The Netherlands, in 2011.

He then worked with the National Institute of Standards and Technology, Gaithersburg, MD, USA. He is currently a Professor with Hunan University, Changsha, China. His research focuses interdisciplinary research, including energy harvesting and edge computing in IoT and power semiconductor devices in microelectronics.



Haibo Zeng (Member, IEEE) received the Ph.D. degree in EECS from the University of California at Berkeley, Berkeley, CA, USA, in 2008.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA. His research interests include embedded systems, cyber–physical systems, and real-time systems.

Dr. Zeng has served as the TPC Member for RTSS, DAC, EMSOFT, RTAS, and ICCAD, and is an Associated Editor for the *ACM Transactions on Embedded Computing Systems* and other journals.

Renfa Li (Senior Member, IEEE) received the Ph.D. degree in electrical engineering and computer sciences from the Huazhong University of Science and Technology, Wuhan, China, in 2002.

He is a Professor with Hunan University, Changsha, China, and was the Dean of the College of Computer Science and Electronic Engineering, Hunan University and the Director of the Key Laboratory for Embedded and Network Computing of Hunan Province. His major research includes embedded systems, distributed systems, and cyber–physical systems.

Di Wu (Member, IEEE) received the Ph.D. degree in computer science from the University of California, Irvine, Irvine, CA, USA, in 2013.

He is currently a Professor with Hunan University, Changsha, China, and an Adjunct Researcher with the University of California, Irvine. His research interests include future networking, intelligent analytics, and smart architecture.

Prof. Wu has actively served on many conference committees and is currently an Associate Editor for the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.