Models on the Move: Towards Feasible Embedded AI for Intrusion Detection on Vehicular CAN Bus

He Xu, Di Wu, Yufeng Lu, and Jiwu Lu, *Hunan University and ExponentiAI Innovation;* Haibo Zeng, *Virginia Tech*

https://www.usenix.org/conference/atc24/presentation/xu-he

This paper is included in the Proceedings of the 2024 USENIX Annual Technical Conference.

July 10–12, 2024 • Santa Clara, CA, USA

978-1-939133-41-0

Models on the Move: Towards Feasible Embedded AI for Intrusion Detection on Vehicular CAN Bus

He Xu[†§], *Di Wu*[†§✉], *Yufeng Lu*[†§], *Jiwu Lu*[†§✉], *Haibo Zeng*[‡]
[†]*Hunan University, China* [‡]*Virginia Tech, USA* [§]*ExponentiAI Innovation*
[✉]*Corresponding Authors: Di Wu (dwu@hnu.edu.cn) and Jiwu Lu (jiwu_lu@hnu.edu.cn)*

Abstract

Controller Area Network (CAN) protocol is widely used in vehicles as 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. To achieve efficient and effective intrusion detection system (IDS) design for hardware and embedded system security in vehicles, we have specifically tackled the challenge that existing IDS techniques rarely consider attacks with small-batch. We propose a model with hardware implementation to function in the vehicular CAN bus, namely MULSAM which employing multi-dimensional long short-term memory with the self-attention mechanism. The self-attention mechanism can enhance the characteristics of CAN bus-oriented attack behavior and the multi-dimensional long short-term memory can effectively extract the in-depth features of time series data. The MULSAM model has been compared with other baselines on five attacks generated by extracting benign CAN data from the actual vehicle. Our experimental results demonstrate that MULSAM has the best training stability and detection accuracy (98.98%) to identify small-batch injection attacks. Furthermore, to speed up the inference of MULSAM as an embedded unit in vehicles, hardware accelerator has been implemented on FPGA to achieve a better energy efficiency than other embedded platform. Even with a certain degree of quantification, the acceleration model for MULSAM still presents a high detection accuracy of 98.81% and a low latency of 1.88 ms, leading to a new cyber-physical system security solution towards feasible embedded AI for intrusion detection on vehicular CAN bus.

1 Introduction

Controller area network (CAN) bus protocol has been widely used in the industrial automation control system due to its low cost, high reliability, real-time, and robust anti-interference ability [14, 43]. In effect, the CAN bus has become a communication standard in the automotive field [1]. The electronic control units (ECUs) perform identity authentication through the CAN ID of the CAN data frame on the CAN bus [24]. However, CAN ID can be arbitrarily modified, which gives an opportunity for intruders to attack the network [13]. 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. In short, due to the communication characteristics of the CAN protocol, the CAN bus as a type of embedded networked sensor systems has inevitable safety hazards, such as illegal control, data leakage, and so on.

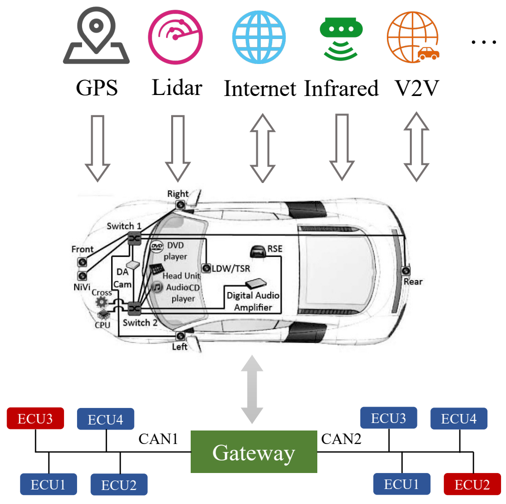


Figure 1: Vehicular communication network architecture.

1.1 Motivations

The automotive embedded system is the central system of the vehicle, as shown in Fig. 1, built based on CAN bus communication network and control the state of the vehicle body by sending and receiving various commands. Intrusion Detection Technology is widely used on the in-vehicle CAN bus [18]. In recent years, with the growth of Intelligent Connected Vehicles (ICVs), 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 [22, 39]. However, few researchers have designed efficient and effective models for attacks with small-batch characteristics. In addition, with the agreement that ICVs is a typical cyber-physical system, practical intrusion detection and prevention deployment on embedded platform to tackle hardware acceleration and hardware security at the same time is in need as well [23, 32].

Meanwhile, we notice that the protein structure prediction through deep learning in gene sequence has achieved tremendous progress and success in recent years. For example, the RoseTTAFold model [2] with a three-track neural network structure has a super high accuracy on protein structure prediction, where the flow of information between different dimensions allows the network to focus on the chemical part of the protein and its folding structure. Inspired by the idea of multi-dimensions in RoseTTAFold, we get the intuition to design our Intrusion Detection System (IDS) [35, 40] based on Multi-dimensional Long Short-Term Memory (MD-LSTM), which can deploy LSTM cells along any or all of the dimensions. To speed up the model inference as an embedded unit in vehicles from practical perspective, Field Programmable Gate Array (FPGA) platform is adopted for hardware accelerator and deployment [6], where the IDS model is sepcifically created in two dimensions. At the same time, to compensate for the loss caused by the reduction of dimensions, the fusion of the self-attention mechanism (SAM) [37] is utilized to improve the detection performance. By doing so, the IDS model can learn multi-dimensional 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, the IDS model can better separate data, which has more complex features and more separated inter-dependencies than standard MD-LSTM networks.

1.2 Challenges

With the unceasing improvement of the automotive intelligence level, the number of in-vehicle network's 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 [12, 17, 38]. The exposed interfaces, such as GPS, V2V, 4G/5G, and so on, have imported many unpredictable security threats to the automobile [8, 41]. 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 [21, 34]. 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. Wherefore, 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 intelligent connected vehicles (ICVs) is crucial imporant [19]. Through the intrusion detecting technology, the vehicle generates an alarm message and switches into a safe protection mode [25]. However, the widely-used automotive embedded systems have limited hardware computing resources due to cost constraints [29]. 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 [7, 11, 36]. The advantage of this method is that no change in the hardware architecture is needed. However, to ensure the safety of vehicles, intrusion detection systems 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.

1.3 Our Contributions

We propose a compact and novel deep learning model with hardware implementation to function as efficient and effective intrusion detection systems in ICVs, namely MULSAM which employing multi-dimensional long short-term memory (MD-LSTM) with the self-attention mechanism (SAM). MULSAM is designed to improve the performance of IDS under multiple different attacks on the vehicle's CAN bus. Its major 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 to optimize network performance. The multi-dimensional concept and the self-attention mechanism 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 be distinguished, which can be regarded as an encoding process. It can enhance MD-LSTM cells' depth and temporal computation, which is capable of processing multi-dimensional 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 deep learning and the data flow-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 self-attention mechanism 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.

2 Related Works

From the perspective of detection approaches, the design of IDS can be divided into Rule-Based and Machine Learning (ML)-Based categories.

2.1 Rule-Based IDS

The rule-based IDS is a simple system, which uses some internal logic relationships of CAN data to perform anomaly analysisn. For instance, Hoppe et al.[10] 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. [31] 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 [20] 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 simple, the algorithm has limited capability to detect attacks with small-batch. Cho and Shin[3] extracted and estimated transmitter's clock skews, which are fingerprints of transmitter 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 [4].

2.2 Machine Learning-Based IDS

The ML-Based IDS is able to process more complex data with multi-dimensional features, but it is also harder to train than Rule-Based IDS besides posing higher deployment cost. BTMonitor [42] 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 [4] extracted the essential features from the message signal and used the multi-class 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 [7] proposed an unsupervised learning approach which combined with 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. [11] 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, spoofing attacks, but did not consider the delete attack that disables a vehicular function, which is common in vehicle bus-attack. Xie et al. [36] proposed an enhanced GAN network to detect whether elaborate CAN message blocks are threatened by data tampering. However, the deep learning GAN model is unable to identify the concrete type of attack and is difficult to train.

Note. Since the vehicular electronic system is in a mobile state during driving, there are strict restrictions on the energy consumption of hardware resources, which brings challenges to deploy a ML-based IDS into the actual vehicular system. In addition, once the functionality of the CAN bus is cracked, the intruder can launch different types of attack with small-batch, which are characterized by their small attack scale and concealed attack patterns, posing difficulty to identify and a significant threat to the driving safety. Therefore, *the power consumption and the detection of small-batch attacks are vital issues for vehicular IDS, whereas the previous works fail to address from the embedded system perspective as a holistic solution.* Correspondingly, we present an enhanced ML-based

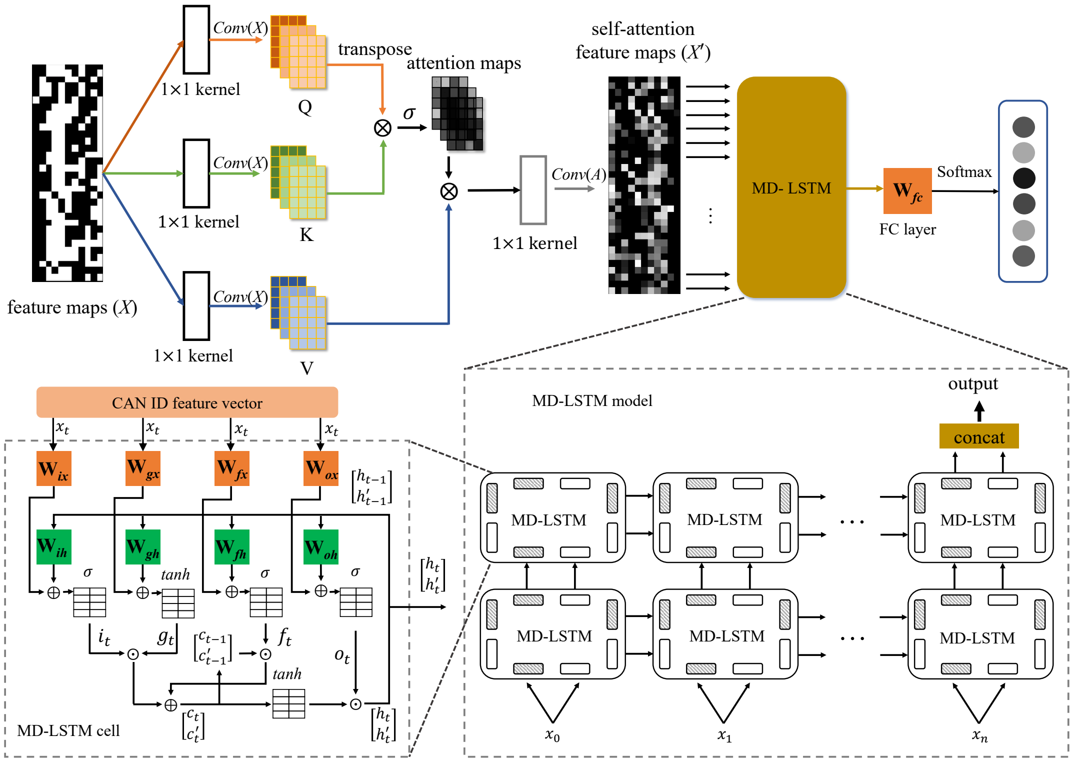


Figure 2: The architecture of MULSAM.

IDS with efficient design and implementation on the FPGA platform for actual vehicular system.

3 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 self-attention mechanism is added to strengthen the correlation between time series data. As shown in Fig. 2, 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. Thirdly, 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. 2, it is essential to notice that the MULSAM proposed in this paper utilizes the self-attention mechanism for making 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 machine learning models.

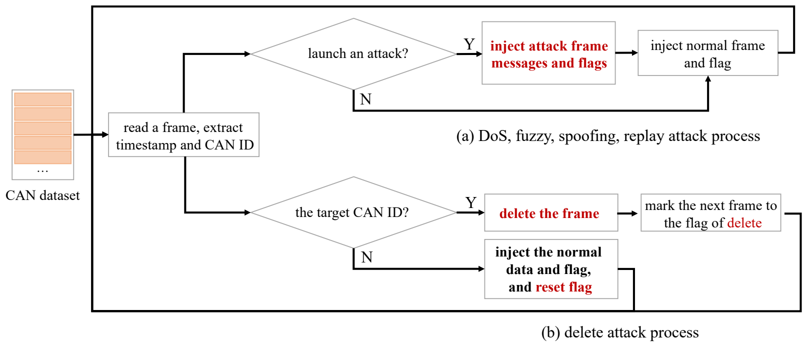
4 Attack Model and Datasets

This paper focuses on five types of attack: 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.

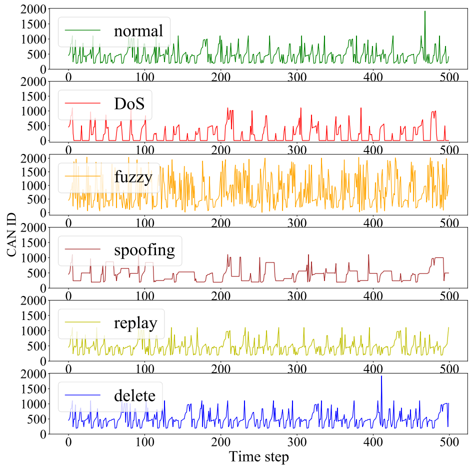
DoS attack. Once attackers invade the CAN bus network, and they will continue to send the highest priority CAN data frame to the CAN bus network, thus occupying the data transmission time window of other normal ECUs. This results in the CAN bus being in a congested state, which may cause the in-vehicle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

Fuzzy attack. The attackers can quickly launch the fuzzy attack without knowing the specific information of the ECUs on the CAN bus of the vehicle system. The difference between 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 to coincide 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


CAN dataset
read a frame, extract timestamp and CAN ID
launch an attack?
Y
N
inject attack frame messages and flags
inject normal frame and flag
(a) DoS, fuzzy, spoofing, replay attack process
the target CAN ID?
delete the frame
mark the next frame to the flag of delete
inject the normal data and flag, and reset flag
(b) delete attack process

(a) The generation process of five attacks



(b) The distribution of attacks' CAN ID

Figure 3: Attack Generation.

eavesdrops on CAN message and then injects many of the same CAN data, which drive 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.

4.1 Generation of Attack

We uses the open-source CAN bus dataset from the 4TU.ResearchData (an international data repository for science, engineering, and design) [5]. The dataset was collected from the actual car while driving. The five types of attack CAN data are generated as shown in Fig. 3 (a) . First, a frame message is read from the original dataset, and the timestamp and CAN ID are extracted. For DoS, fuzzing, spoofing and replay attacks, the time stamp of the first frame of data is used as the starting time to calculate the time of launching the attack, and compare the attack time with the current extracted time stamp to determine whether to launch an attack. If an attack is initiated, an attack frame and an attack flag are injected. If no attack is initiated, a normal frame and a normal flag are injected. For delete attack, it is only necessary to judge whether the current extracted CAN ID is the target and whether to launch an attack. If an attack is launched, delete the frame and mark the next message as a delete attack. If no attack is launched, normal frame and normal marker are injected. The flags of 0, 1, 2, 3, 4, 5 represent normal state, DoS attack, fuzzing attack,spoofing attack, replay attack and delete attack, respectively. Through the above attack methods, this paper generates five anomalous datasets with attack characteristics. Fig. 3 (b) shows the distribution of CAN ID characteristics in the normal state and in five attack datasets.

From Fig. 3 (b), the difference between the characteristics of DoS, fuzzy, and spoofing attack with the normal data state can be seen clearly due to their flooding features. However, the characteristics of replay and delete attack are not evident, which means that small-batch attack is more difficult to distinguish because it is similar to the benign state. The experiment in Section 6 also proves this assumption.

4.2 Preprocessing

The input data usually need 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. Eq. (1) can calculate the 11bit features in the original CAN ID data.

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

where *can_id* represents the original CAN ID, i represents the bit position to be extracted, x_i is the i-th 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.

5 MULSAM Design

This paper proposes a MD-LSTM architecture with the front Self-Attention Mechanism (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 [30], is utilized to enhance the correlation between the time-series data and easily 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.

5.1 Neural Network Design

5.1.1 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. 4 (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 in our MULSAM is replaced with the convolutional network, and the Softmax activation function is modified to the logic sigmoid activation function as shown in Fig. 4 (right). This scheme also benefits the model algorithm design based on the FPGA platform to execute better parallelly.

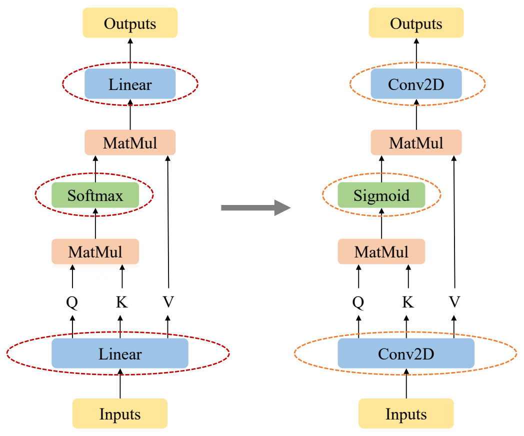


Figure 4: The typical self-attention (left) and its enhanced architecture (right).

First, the input data is defined as $X = (x_0, x_1, x_2, ..., x_n)$, 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.

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

$$(Q, K, V) = \text{Conv2D}(X, (1, 1), Hidden\_Dim * 3) \quad (2)$$

$$\tilde{A} = \sigma(KQ^T) \quad (3)$$

$$\text{Attn}(\tilde{A}, V) = \text{Conv2D}(\tilde{A}V, (1, 1), Output\_Dim) \quad (4)$$

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 Eq. (2) is dimensional evenly divided into three parts, namely Q, K, V. Then, the attention maps are calculated after activation through Eq. (3), and the output of the self-attention layer is got according to Eq. (4). 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.

5.1.2 MD-LSTM in MULSAM

Unlike the traditional Stacked LSTM, the MD-LSTM network [16], as shown in Fig. 5, 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.

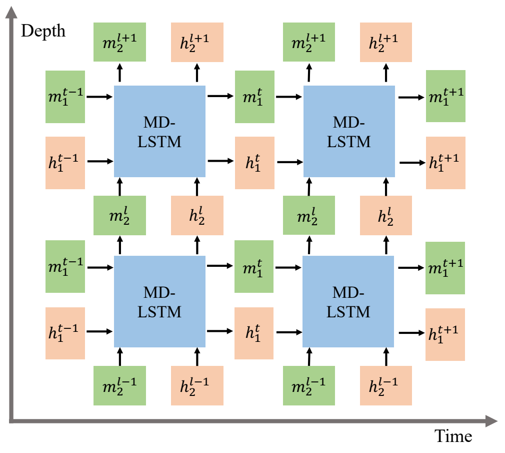


Figure 5: The architecture of MD-LSTM.

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 paper, and the MD-LSTM cell can be split into two LSTM cells.

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

$$\begin{aligned} g_t &= \tanh(W_{gx}x_t + W_{gh}h_{t-1} + b_g) \\ i_t &= \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \\ f_t &= \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \\ o_t &= \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \\ c_t &= g_t \odot i_t + c_{t-1} \odot f_t \\ h_t &= \tanh(c_t) \odot o_t \end{aligned} \tag{5}$$

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 Eq. (5) as follows:

$$(h_t, c_t) = \mathrm{LSTM}(x_t, h_{t-1}, c_{t-1}, W_i, W_h) \tag{6}$$

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:

$$\begin{aligned} (h_t^1, c_t^1) &= \mathrm{LSTM}(x_t, h_{t-1}^1, c_{t-1}^1, W_i^1, W_h^1) \\ (h_t^2, c_t^2) &= \mathrm{LSTM}'(x_t, h_{t-1}^2, c_{t-1}^2, W_i^2, W_h^2) \end{aligned} \tag{7}$$

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:

$$H = \left[h_N^1, h_N^2\right] \tag{8}$$

where N in Eq. (8) is the total number of time steps.

5.2 FPGA-based Model Design

How to improving the data locality of matrix structures is a crucial problem for maximizing the performance of the machine learning 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[28]. To implement MULSAM application deployment at the edge device, we analyze the internal parallelism of the algorithm, and the hardware circuit is realized by Vivado high-level synthesis (HLS) [27]. 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.

5.2.1 Self-Attention Pipeline Design

For CPUs or GPUs, each step of the computation of the self-attention layer, as shown in Fig. (4) (right), requires waiting for the completion of the previous step. For example, The KQ^T operation in Eq. (4) can be computed when the Q and K matrices are fully completed. The algorithm needs to be refactored because each module's input and the output stream are FIFO queues instead of a complete matrix.

(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. The Alg. 1 shows the whole process of this module.

Algorithm 1 (Q,K,V) calculation

```
Require: 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
Ensure: Q, K, V
```

Activation calculation module design. Since the Q, K matrices are calculated sequentially, then KQ^T in Eq. (4) cannot be fully calculated at one time. The calculation expression of the element of KQ^T is as follows:

$$a_{i,m} = \sum_{j=0}^{11 \times N - 1} k_{i,j} q_{j,m} \quad i, m \in (0, 1, \ldots, H-1) \tag{9}$$

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 Eq. 10, the $q_{j,i}$ and $k_{i,j}$ are also used to get the product with the cached K and Q elements, respectively. And 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.

$$\begin{aligned} a_{z,i} += k_{z,j} q_{j,i}, \quad & z \in (0,1,...,i-1) \\ a_{i,w} += k_{i,j} q_{j,w}, \quad & w \in (0,1,...,i-1) \end{aligned} \tag{10}$$

The essence of Eq. (10) is to split Eq. (9) 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 Eq. (3) can not be computed unless obtaining all data in the Q and K FIFO queues. The Alg. 2 shows the whole process of this module.

Algorithm 2 Activation calculation

```
Require: 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
Ensure: $\tilde{A} = \sigma(A)$
```

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 Alg. 3, an element $\tilde{a}_{i,j}$ of the current input $\tilde{A}$ is multiplied by the element v_j of the j-th row of V, and then the result is accumulated to the element r_i of the i-th row of the result matrix. The calculation process is shown in Eq. (11).

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

Since the value of the V matrix is completely computed, when $\tilde{A}$ matrix inputs a row of data, a row of data of $\tilde{A}$V can be calculated as Eq. (12).

$$s_{o,i} = W_o \sum_{j=0}^{H-1} r_{i,j} \tag{12}$$

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

Algorithm 3 Output calculation

```
Require: $\tilde{A}, V$
 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
Ensure: $Attn\_out$ matrix
```

5.2.2 MD-LSTM Pipeline Design

Following the same design principles as that of the self-attention mechanism 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.

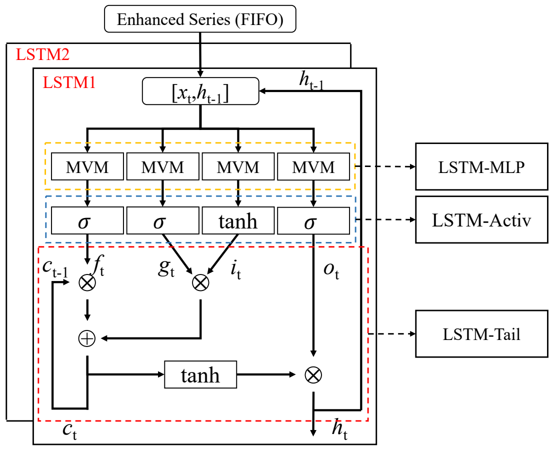


Figure 6: A step computation of LSTM cell.

As shown in Fig. 6, the calculation process of the LSTM cell is restructured into three modules, including LSTM-MLP, LSTM-Activ, and LSTM-Tail. which 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 Alg. 4 to perform the four MVM parallelly.

Algorithm 4 LSTM-MLP calculation

```
Require: $Attn\_out$ 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
Ensure: $gifo$ matrix
```

LSTM-Activ. The LSTM-Activ module is not time-dependent so the activation value can be quickly calculated 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. The Alg. 5 shows the whole process of this module.

Algorithm 5 LSTM-Activ

```
Require: $gifo$ 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
Ensure: $g_t, i_t, f_t, o_t$
```

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 line 3 and line 4 in Alg. 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.

Algorithm 6 LSTM-Tail

```
Require: $i_t, f_t, g_t, o_t$
 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
Ensure: $c_t, h_t$
```

6 Experiments and Evaluation

We first designed and trained different baseline comparison models on the PC. Then the computation architecture of MULSAM was redesigned to make them suitable for deployment on the Ultra96-V2 as an FPGA development 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.

Our evaluation focuses on three aspects of the performance: different model depths, various machine learning models, and two above-mentioned models based on the FPGA platform.

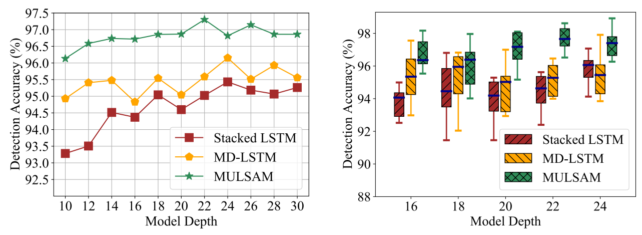
6.1 Different Model Depths

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 self-attention mechanism 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.

Table 1: Model Parameters.

Model Parameters	Stacked LSTM	MD-LSTM	MULSAM
Hidden dimension	24	24	24
Output dimension	6	6	6
Recurrent dimension	2	2	2
Self-attention weight matrices	N/A	N/A	32×11
Number of parameters	1720	2500	4996

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.



(a) Overall Accuracy (b) Trainning Stability

Figure 7: Performance evaluation on different model depth.

Since MULSAM is based on multi-dimensional LSTM (MD-LSTM) with self-attention mechanism, we could compare it with MD-LSTM without self-attention mechanism, and MD-LSTM was also compared with Stacked LSTM, as the ablation study. Our model parameters are specified in Tab. 1.

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. 7 (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.7 (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.

Differences in classification. As shown in Fig. 8, 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 have the situation of identifying 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 multi-dimensional architecture and the self-attention layer in MULSAM. This demonstrates that MULSAM will have a lower false-positive rate and can effectively enhance the characteristics of small-batch attacks.

6.2 Various Machine Learning Models

Our baseline comparison models include SVM, MLP, CNN, Stacked LSTM, MD-LSTM and Transformer. The evaluation metrics of comparison include the overall accuracy, precision, recall, and F_1 score. The details of the metrics are as Eq. (13).

$$\begin{aligned} Accuracy &= \frac{TP+TN}{TP+FN+FP+TN} \\ Precision &= \frac{TP}{TP+FP} \\ Recall &= \frac{TP}{TP+FN} \\ F_1 &= \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \end{aligned} \tag{13}$$

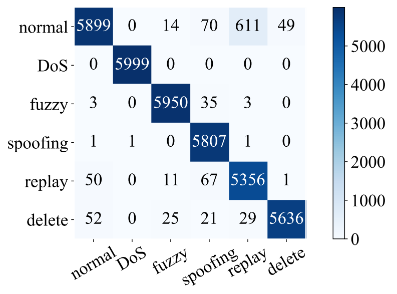
Where TP, FP, TN, and FN are four outcomes of the classification, representing True Positive, False Positive, True Negative, and False Negative, respectively.

Table 2: Model Initial Hyper-parameters

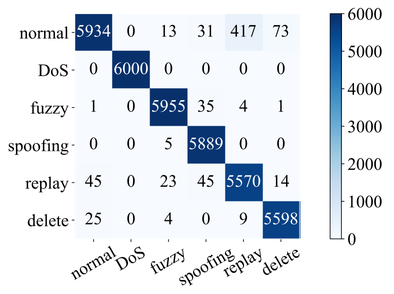
Parameters	**Value**
Epochs	100
Early stopping	5
Activation Function	Softmax
Learning rate	1e-3
optimizer	Adam
Loss Function	CrossEntropyLoss
Batch size	128
Steps	32

The cross-entropy loss function, defined as $H(p,q) = -\sum_x p(x)\log q(x)$, is used as the loss function for all machine learning 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 Tab. 2 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 hyper-parameter debugged and selected according to the validation set. The adaptive gradient algorithm (Adam) [15], which can adjust the learning rate, is used to optimize our models. To prevent the over-fitting and to reduce the complexity of these models, the dropout technique [26] is used in all deep learning models. The total steps of the input series is set to 32, which corresponds to approximately 30ms of CAN messages.

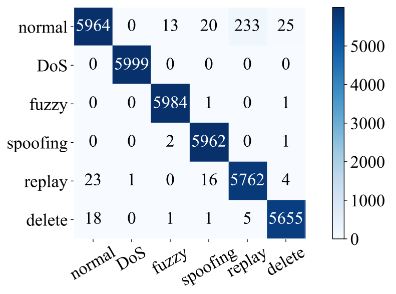
The performance of various models has been shown in Tab. 3. As we can see, the SVM, CNN, and MLP perform well in detecting attacks with flooding properties but perform poorly on normal state and attacks with small-batch, which cause the overall accuracy to be less than 90%, while that of LSTM-related models is more than 97%. For example, since the feature distribution of normal CAN data and small-batch attack is very close, the recall of SVM, MLP, CNN are only 26.25%, 54.10%, and 67.11%, respectively. The superior performance



(a) Stacked LSTM



(b) MD-LSTM



(c) MULSAM

Figure 8: Confusion matrix of classification.

Table 3: Performance on Various Models.

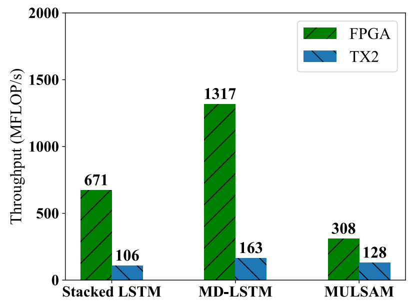
Model	**Accuracy** (%)	**Attack**	**Recall**	**Precision**	F_1
SVM	82.07	normal	0.2623	0.1456	0.1008
		DoS	0.9983	0.9953	0.9923
		fuzzy	0.9439	0.9650	0.9870
		spoofing	0.9814	0.9272	0.8787
		replay	0.7428	0.7023	0.6660
		delete	0.5968	0.6909	0.8202
MLP	80.08	normal	0.5410	0.7795	0.6387
		DoS	0.9988	0.9463	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
Transformer	98.26	normal	0.9273	0.9906	0.9684
		DoS	**1.0000**	0.9998	0.9999
		fuzzy	0.9932	0.9946	0.9951
		spoofing	0.9993	0.9886	0.9934
		replay	**0.9953**	0.9438	0.9556
		delete	**0.9982**	0.9911	0.9903
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**

of LSTM-related models might be that the structure of LSTM is suitable for processing the data with time correlation[9], while the form of SVM, MLP, CNN fail to do. MULSAM has a recall of 95.35% on the detection of the normal state, while 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 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 than other LSTM-based models.

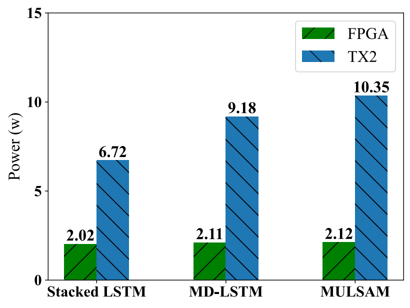
Before we deployed models on FPGA, we also compared MULSAM with Transformer [30], which is a seq2seq model using multi-head attention structures for time series processing. We found that Transformer is better than our MULSAM on recall score for replay attack and delete attack, but MULSAM presents very close performance to Transformer on the metric and better results on other metrics. In addition, MULSAM outperforms Transformer on all metrics for other attack types (DoS, fuzzy, spoofing). Considering that multi-head attention structure to realize the simultaneous input of data in Transformer makes the computational complexity of the model reach $O(n^2)$, and at the same time has a memory bottleneck problem when inputting long sequences, it is hard to run Transformer on the resource-constrained Ultra96-V2 we used as an embedded FPGA development board. In addition, because MULSAM has presented better performance in general than Transformer on PC as shown in Tab. 3, it is also not necessary to compare MULSAM with Transformer on the FPGA platform.

6.3 Performance on Hardware Acceleration

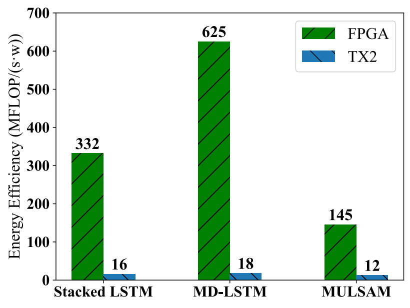
We also present the results of the implementations of three LSTM-related models, including Stacked LSTM, MD-LSTM, and MULSAM. Due to the advantages of the modular design approach, the programming of the LSTM cell in MULSAM



(a) Throughput Rate



(b) Power



(c) Energy efficiency

Figure 9: The comparison of different accelerators between FPGA and TX2 platforms.

Table 4: The Selected Depth and Corresponding Accuracy.

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

can be conveniently reused by Stacked LSTM and MD-LSTM. The model depth of Stacked LSTM, MD-LSTM, and MULSAM are selected corresponding to the highest accuracy in Fig. 7 (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 Tab. 4.

6.3.1 Comparison between FPGA and TX2

To compare FPGA-based embedded system security with GPU-based embedded platform, we designed and trained different baseline models on Jetson TX2 as the Commercial-Off-The-Shelf GPU-based embedded platform. The Jetson TX2 is a 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. 9 (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. 9 (b), the power consumption of all three models exceeds 5 W on the TX2, while that in 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 from Fig. 9 (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.

Therefore, these performance evaluation results on throughput rate, power and energy efficiency have verified that our FPGA-based implementation can achieve practical intrusion detection and prevention deployment on embedded platform to tackle hardware acceleration and hardware security at the same time. The multi-dimensional concept and the self-attention mechanism adopted in MULSAM make it tiny and parallel, with inherent nature tailored for deploying on FPGA platform. In comparison with the mainstream Jetson TX2 which is GPU-based embedded platform and assisted with ARM TrustZone [33], our embedded AI running on FPGA is obviously more efficient and effective for intrusion detection on vehicular CAN bus.

6.3.2 Models on the Move

We conducted testing experiments on an Ultra96-v2 embedded device installed in actual vehicle as an FPGA-enabled gateway for intrusion detection on vehicular CAN bus, as shown in Fig. 10. The embedded experiments for Stacked LSTM, MD-LSTM and MULSAM are run based on their trained models from above-mentioned attack datasets and tested in the real-world driving scenario to guarantee that the vehicle can prevent malicious navigation from small-batch attacks which usually happen in this case, and handle corresponding attacks in short time.

Resource overhead. The resource overhead has been visualized after the model has been synthesized to RTL (Register Transfer Level) 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 two-dimensional LSTM, so the resource overhead

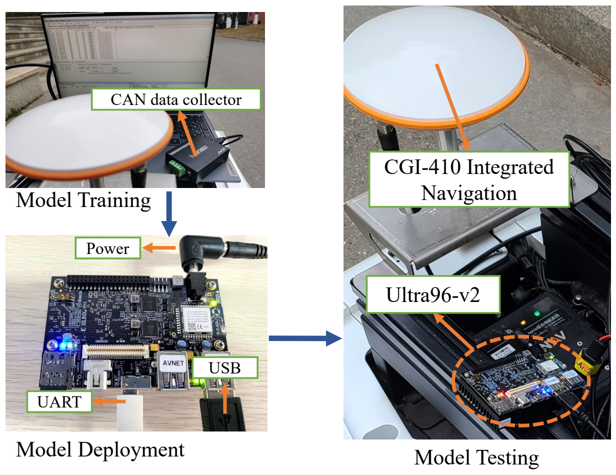
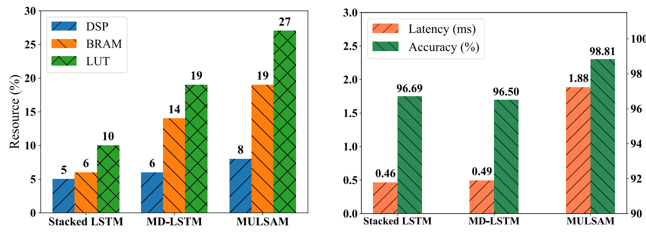


Figure 10: FPGA device during testing.

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 Tab. 4, the fixed-point quantization in the FPGA-based accelerators results 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 multi-dimensional architecture of MD-LSTM causes its performance to be more sensitive to the weight parameters. However, MULSAM also has a multi-dimensional architecture. Still, it has the lowest drop of accuracy (98.81%), which is likely that the self-attention mechanism 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. Specifically, since the step length of our MULSAM model is 32, it can process 32 CAN messages at a time. Given that the calculation time for one iteration of MULSAM is 1.88ms as the maximum duration, and assuming each CAN message is 110 bits in size according to CAN standard, the approximate amount of data that MULSAM can process per second is $1000ms/1.88ms \times 32 \times 110bit = 1.78Mbit$. This translates to a data processing rate of 1.78 Mb/s for MULSAM, which is significantly higher than the rate of a saturated CAN bus (*e.g.* 250 Kb/s to 1 Mb/s). Therefore, our proposed solution can monitor a nearly saturated CAN bus in real time.



(a) Resource Overhead (b) Accuracy and Latency

Figure 11: The performance evaluation on FPGA device.

7 Conclusion

The development of automotive intelligence has also brought more security threats, affecting the benign message transmission of the in-vehicle CAN bus communication network. For this problem, an enhanced intrusion detection technology is developed based on MD-LSTM and Self-Attention Mechanism (MULSAM). First, five attack datasets are generated based on the attack-free time-series data by extracting the CAN ID field in the CAN message. Our proposed model can detect the type of attack with the small-batch, whereas the previous machine learning models fail to do so. Second, to deploy our model on the vehicle edge, the computation process of MULSAM is reconstructed by adopting multiple parallel methods and implemented based on the FPGA platform. The experiment proved that the lightweight reduction of the weights of the recurrent neural network did not impact the detection accuracy. In the future, we will work on a real-time online detection system and run more real-world tests to evaluate the performance of MULSAM accelerator in autonomous driving scenarios. Additional attacks including emerging ones designed for vehicular CAN bus will be verified as well if it is necessary.

Acknowledgments

We thank the anonymous reviewers for their constructive comments and insightful suggestions. This work was partially supported by the National Natural Science Foundation of China under Grant No. 61972145 and 61932010, the Key R&D Program of Hunan Province under Grant No. 2022GK2069, and the Fundamental Research Funds for the Central Universities.

References

[1] Omid Avatefipour and Hafiz Malik. State-of-the-art survey on in-vehicle network communication (can-bus) security and vulnerabilities. *arXiv preprint arXiv:1802.01725*, 2018.

[2] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N. Kinch, R. Dustin

Schaeffer, Claudia Millán, Hahnbeom Park, Carson Adams, Caleb R. Glassman, Andy DeGiovanni, Jose H. Pereira, Andria V. Rodrigues, Alberdina A. van Dijk, Ana C. Ebrecht, Diederik J. Opperman, Theo Sagmeister, Christoph Buhlheller, Tea Pavkov-Keller, Manoj K. Rathinaswamy, Udit Dalwadi, Calvin K. Yip, John E. Burke, K. Christopher Garcia, Nick V. Grishin, Paul D. Adams, Randy J. Read, and David Baker. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

[3] Kyong-Tak Cho and Kang G Shin. Fingerprinting electronic control units for vehicle intrusion detection. In *25th USENIX Security Symposium (Security)*, pages 911–927, 2016.

[4] Wonsuk Choi, Kyungho Joo, Hyo Jin Jo, Moon Chan Park, and Dong Hoon Lee. Voltageids: Low-level communication characteristics for automotive intrusion detection system. *IEEE Transactions on Information Forensics and Security*, 13(8):2114–2129, 2018.

[5] Guillaume Dupont, Alexios Lekidis, J. (Jerry) den Hartog, and S. (Sandro) Etalle. Automotive controller area network (can) bus intrusion dataset v2, Nov 2019.

[6] Ilias Giechaskiel, Kasper Bonne Rasmussen, and Jakub Szefer. C3apsule: Cross-fpga covert-channel attacks through power supply unit leakage. In *IEEE Symposium on Security and Privacy*, pages 1728–1741, 2020.

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

[8] Yuze He, Li Ma, Jiahe Cui, Zhenyu Yan, Guoliang Xing, Sen Wang, Qintao Hu, and Chen Pan. Automatch: Leveraging traffic camera to improve perception and localization of autonomous vehicles. In *ACM SenSys*, pages 16–30, 2022.

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

[10] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Applying intrusion detection to automotive it-early insights and remaining challenges. *Journal of Information Assurance and Security (JIAS)*, 4(6):226–235, 2009.

[11] Md Delwar Hossain, Hiroyuki Inoue, Hideya Ochiai, Doudou Fall, and Youki Kadobayashi. Lstm-based intrusion detection system for in-vehicle can bus communications. *IEEE Access*, 8:185489–185502, 2020.

[12] Tianxiang Huang, Jianying Zhou, and Andrei Bytes. Atg: An attack traffic generation tool for security testing of in-vehicle can bus. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–6, 2018.

[13] Hyo Jin Jo, Jin Hyun Kim, Hyon-Young Choi, Wonsuk Choi, Dong Hoon Lee, and Insup Lee. Mauth-can: Masquerade-attack-proof authentication for in-vehicle networks. *IEEE transactions on vehicular technology*, 69(2):2204–2218, 2019.

[14] Karl Henrik Johansson, Martin Törngren, and Lars Nielsen. Vehicle applications of controller area network. In *Handbook of networked and embedded control systems*, pages 741–765. Springer, 2005.

[15] Pooria Joulani, Andras Gyorgy, and Csaba Szepesvári. Delay-tolerant online convex optimization: Unified analysis and adaptive-gradient algorithms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[16] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.

[17] Kyounggon Kim, Jun Seok Kim, Seonghoon Jeong, Jo-Hee Park, and Huy Kang Kim. Cybersecurity for autonomous vehicles: Review of attacks and defense. *Computers & Security*, page 102150, 2021.

[18] Sekar Kulandaivel, Shalabh Jain, Jorge Guajardo, and Vyas Sekar. Cannon: Reliable and stealthy remote shutdown attacks via unaltered automotive microcontrollers. In *IEEE Symposium on Security and Privacy*, pages 195–210, 2021.

[19] Shuheng Li, Ranak Roy Chowdhury, Jingbo Shang, Rajesh K. Gupta, and Dezhi Hong. Units: Short-time fourier inspired neural networks for sensory time series classification. In *ACM SenSys*, pages 234–247, 2021.

[20] Congli Ling and Dongqin Feng. An algorithm for detection of malicious messages on can buses. In *2012 national conference on information technology and computer science. Atlantis Press*, volume 10. Citeseer, 2012.

[21] Guifu Ma, Manjiang Hu, Xiaowei Wang, Haoran Li, Yougang Bian, Konglin Zhu, and Di Wu. Joint partial offloading and resource allocation for vehicular federated learning tasks. *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[22] Minghua Ma, Shenglin Zhang, Junjie Chen, Jim Xu, Haozhe Li, Yongliang Lin, Xiaohui Nie, Bo Zhou, Yong Wang, and Dan Pei. Jump-starting multivariate time series anomaly detection for online service systems. In *USENIX ATC 2021*, pages 413–426, 2021.

[23] Ratul Mahajan, Jitendra Padhye, Sharad Agarwal, and

Brian Zill. High performance vehicular connectivity with opportunistic erasure coding. In *USENIX ATC 2012*, pages 237–248, 2012.

[24] Mirco Marchetti and Dario Stabili. Anomaly detection of can bus messages through analysis of id sequences. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1577–1583. IEEE, 2017.

[25] Ishtiaq Rouf, Robert D Miller, Hossen A Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe, and Ivan Seskar. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *USENIX Security Symposium*, volume 10, 2010.

[26] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems*, 29:1163–1171, 2016.

[27] Benjamin Carrion Schafer and Zi Wang. High-level synthesis design space exploration: Past, present, and future. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2628–2639, 2020.

[28] Marco Siracusa and Fabrizio Ferrandi. Tensor optimization for high-level synthesis design flows. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4217–4228, 2020.

[29] Rebecca Smith and Scott Rixner. Surviving peripheral failures in embedded systems. In *USENIX ATC 2015*, pages 125–137, 2015.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

[31] Tuan Phan Vuong, George Loukas, and Diane Gan. Performance evaluation of cyber-physical intrusion detection on a robotic vehicle. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 2106–2113. IEEE, 2015.

[32] Jie Wang, Yuewu Wang, Lingguang Lei, Kun Sun, Jiwu Jing, and Quan Zhou. Trustict: an efficient trusted interaction interface between isolated execution domains on arm multi-core processors. In *ACM SenSys*, pages 271–284, 2020.

[33] Jinwen Wang, Ao Li, Haoran Li, Chenyang Lu, and Ning Zhang. Rt-tee: Real-time system availability for cyber-physical systems using arm trustzone. In *IEEE Symposium on Security and Privacy*, pages 352–369, 2022.

[34] Di Wu, Qiang Liu, Yong Li, Julie A. McCann, Amelia C. Regan, and Nalini Venkatasubramanian. Adaptive lookup of open wifi using crowdsensing. *IEEE/ACM Transactions on Networking*, 24(6):3634–3647, 2016.

[35] Di Wu, He Xu, Zhongkai Jiang, Weiren Yu, Xuetao Wei, and Jiwu Lu. Edgelstm: Towards deep and sequential edge computing for iot applications. *IEEE/ACM Transactions on Networking*, 29(4):1895–1908, 2021.

[36] Guoqi Xie, Laurence T Yang, Yuanda Yang, Haibo Luo, Renfa Li, and Mamoun Alazab. Threat analysis for automotive can networks: A gan model-based intrusion detection technique. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[37] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[38] Rui Zhang and Cynthia Sturton. Transys: Leveraging common security properties across hardware designs. In *IEEE Symposium on Security and Privacy*, pages 1713–1727, 2020.

[39] Yan Zhang, Yi Zhu, Zihao Liu, Chenglin Miao, Foad Hajiaghajani, Lu Su, and Chunming Qiao. Towards backdoor attacks against lidar object detection in autonomous driving. In *ACM SenSys*, pages 533–547, 2022.

[40] Youqian Zhang and Kasper Rasmussen. Siraj: A unified framework for aggregation of malicious entity detectors. In *IEEE Symposium on Security and Privacy*, pages 507–521, 2022.

[41] Zhi Zhang, Yueqiang Cheng, Minghua Wang, Wei He, Wenhao Wang, Surya Nepal, Yansong Gao, Kang Li, Zhe Wang, and Chenggang Wu. Softtrr: Protect page tables against rowhammer attacks using software-only target row refresh. In *USENIX ATC 2022*, pages 399–414, 2022.

[42] Jia Zhou, Prachi Joshi, Haibo Zeng, and Renfa Li. Btmonitor: Bit-time-based intrusion detection and attacker identification in controller area network. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(6):1–23, 2019.

[43] Yi Zhu, Chenglin Miao, Foad Hajiaghajani, Mengdi Huai, Lu Su, and Chunming Qiao. Adversarial attacks against lidar semantic segmentation in autonomous driving. In *ACM SenSys*, pages 329–342, 2021.