Received: 22 June 2022 | Accepted: 6 August 2022

DOI: 10.1002/int.23012

RESEARCH ARTICLE

WILEY

STC-IDS: Spatial–temporal correlation feature analyzing based intrusion detection system for intelligent connected vehicles

Pengzhou Cheng[1,2] | Mu Han[1] | Aoxue Li[3] | Fengwei Zhang[4]

[1]School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China

[2]School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai, China

[3]School of Automotive and Traffic Engineering, Jiangsu University, Zhenjiang, China

[4]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China

Correspondence
Mu Han, School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China.
Email: hanmuktz888@gmail.com

Funding information
Natural Science Fund for Colleges and Universities in Jiangsu Province, Grant/Award Number: 12KJD580002; Jiangsu Graduate Innovation Fund, Grant/Award Number: KYLX1057; Key Research and Development Plan of Jiangsu Province in 2017 (Industry Foresight and Generic Key Technology), Grant/Award Number: BE2017035

Abstract

Intrusion detection is an important defensive measure for automotive communications security. Accurate frame detection models assist vehicles to avoid malicious attacks. Uncertainty and diversity regarding attack methods make this task challenging. However, the existing works have the limitation of only considering local features or the weak feature mapping of multi-features. To address these limitations, we present a novel model for automotive intrusion detection by spatial–temporal correlation (STC) features of in-vehicle communication traffic (intrusion detection system [IDS]). Specifically, the proposed model exploits an encoding-detection architecture. In the encoder part, spatial and temporal relations are encoded simultaneously. To strengthen the relationship between features, the attention-based convolutional network still captures spatial and channel features to increase the receptive field, while attention-long short-term memory builds meaningful relationships from previous time series or crucial bytes. The encoded information is then passed to detector for generating forceful spatial–temporal attention features and enabling anomaly classification. In particular, single-frame and multiframe models are constructed to present different advantages, respectively. Under automatic hyperparameter selection based on

Bayesian optimization, the model is trained to attain the best performance. Extensive empirical studies based on a real-world vehicle attack data set demonstrate that STC-IDS has outperformed baseline methods and obtains fewer false-alarm rates while maintaining efficiency.

KEYWORDS

attention mechanism, control area network, intrusion detection system, in-vehicle networks, spatial–temporal features

1 | INTRODUCTION

Nowadays, a large number of electronic control units (ECUs) have replaced the mechanical control units to manage the assorted functions of in-vehicle control systems. The ECUs are interconnected to exchange varied vehicle information with each other via networks referred to as in-vehicle networks (IVNs), such as controller area networks (CANs).[1] Along with local interconnected network LIN and FlexRay, CAN is well known and most employed as the defactor standard for IVNs.[2,3] It is noteworthy that CAN was developed as a broadcast-based communication protocol that supports the maximum baud rate up to 1 Mbps. Furthermore, the fault-tolerant detection mechanism guarantees the stability of message transmission.

However, the CAN bus is potentially vulnerable to attacks owing to the lack of security mechanisms, such as encryption, access control, and authentication.[4–6] Cyber security is becoming a major concern for IVNs systems as increasingly more security researchers demonstrate their ability to launch attacks on actual vehicles.[7] What can be investigated is various attacks have been threatening several significant components of IVNs.[4,8,9] For instance, 360 cyberattack Lab adopted electronic radio-frequency technology to successfully hack into Tesla in 2015.[10] Miller et al. invaded the Jeep Cherokee's IVNs system using an open Wi-Fi port and reprogrammed the firmware of ECU. They succeeded in taking control of a wide range of vehicle functions (e.g., disabling the brakes and stopping the engine), triggering a recall of 1.4 million vehicles.[4] Thereafter, the electric features lift, warning lights, airbag, and tire pressure monitoring system have also become the target of attack.[11] Incredibly, these attacks have multiple ways of being performed. As such, the study on the security of IVNs is attracting significant attention from security researchers.[12,13]

There are many available methods that have the ability to protect IVNs secure, where IDS as an effective defense method has attracted more attention from researchers.[14,15] Currently, a host of IDS schemes is rule-based and statistical-based. Although accuracy and efficiency are excellent about some common attacks, the passive characteristic and the constant need for updates result in a certain restriction.[16] With the increase in vehicle computing power and the maturity in machine learning (ML) technology, they promote the further development of IDS.[17] Real-time, higher detection, and lower false-positive rates have been a fervent research problem for deep learning (DL)-based in-vehicle IDS. Simultaneously, inadequate feature extraction, complex network structure, and more parameters, also are pending breakthroughs.[1,18] To address such limitations, variants of IDS based on DL have been proposed in recent years.[19,20] However, these variants merely consider the partial features, either time series of CAN intrusion detection (ID) or CAN data field. Moreover, spatial–temporal correlation (STC) features have been shown to better capture the details of the message in anomaly detection,[21–23] whereas how to create and represent the relationship between spatial–temporal features becomes a vital concern to elevate detection performance in the automotive intrusion.[24,25]

Thus, the purpose of this paper is to provide STC features analyzing based-enhanced intrusion detection system (STC-IDS). On the basis of an encoding-detection architecture, the intuition first trained a boosted convolutional long short-term memory (LSTM) parallel feature extraction model. Improved attention-based LSTM network (A-LSTM) captures the temporal features and builds important relationships from previous sequences or crucial bytes. Meanwhile, a reduced VGGNet network can learn the spatial features from CAN frames. Moreover, the attention convolutional block (A-Conv2D) enables attaining a broad perspective through multichannel features to refine feature mapping. Unlike many previous methods, it can concentrate more on changes in crucial areas and ignore bytes that are regular and unchanging. Afterward, STCs between features are established to feed into the detector as a two-class classification problem. Note that both single-frame and multiframe models are considered in this paper to present different advantages, respectively.

This paper makes the following contributions.

1. On the basis of analyzing STC features of CAN messages in detail, we propose an enhanced convolutional LSTM spatial–temporal feature encoder with attention. The single-frame model automatically captures the important byte relationships of each CAN frame and uses convolutional components to find where key bytes are. Moreover, the multiframe model captures significant relationships from previous time series, in which the attention convolutional block, assisted by spatial attention and channel attention, is able to snap changes in crucial areas.
2. Further, the detector achieves anomaly detection for the constructed representative STC features after multiview learning. We evaluated the detection performance of our scheme using a publicly available real-vehicle CAN data set. We also compare it with the baseline model and show a significant improvement in detection performance and a reduction in false-positive rates and error rates.
3. By performing the injection attack in the same way, we calculated the detection efficiency of the model on real vehicles. The multiframe model has sufficient ability to satisfy real-time detection. In addition, the single-frame model combined with database retrieval has the ability to trace anomalous ECUs.

The rest of this paper is organized as follows. Section 2 presents a discussion of related work on CAN-based IDS. Section 3 introduces the CAN bus protocol, vulnerability, and analyzes the CAN frame with spatial–temporal. Section 4 proposes the parallel network model based on spatial–temporal features analysis. The experiment result of the proposed model on the public real-vehicle data set is described in Section 5. Finally, we conclude this study and look ahead at in-vehicle IDS potential perspectives.

2 | RELATED WORK

In this section, we provide an in-depth discussion about the research situation in anomaly detection and ID for in-vehicle CAN communication systems. They are divided into four categories, namely, specification-, fingerprint-, statistical-, and ML-based approaches, as summarized in Table 1.

TABLE 1 Comparison of in-vehicle intrusion detection systems

Categories	Research work	Technique	Evaluation data	Contribution	Limitation
Specification based	Studnia et al.[26]	A list of signatures derived from CAN data set	Real	Automatically generating a set of forbidden sequences	The length of CAN bus words may not be known a priori
	Dagan and Wool[27]	An antispoofing system detects CAN message IDs by each ECU that are not sent by the ECU itself	Real and simulation	Effective resistance to spoofing attacks	Each ECU takes on the IDS role, adding a certain burden to communication
	Olufowobi et al.[20]	Worst-case response time analysis	Real and synthesized	Real-time parameter estimation algorithm developed in a black-box approach	High false-positive rate and relatively low performance on open data set
Fingerprint based	Cho and Shin[28]	Analyzed the clock skew of ECUs	Real and simulation	Proposed an ECU profiling method according to electrical signal characteristics	Workable only to periodic messages excluding nonperiodic messages
	Choi et al.[29]	Established the electrical signal features of each ECU based on time domain and frequency domain	Real and simulation	Effective of making a distinction between ECU malfunctions and a bus-off attack	The electrical properties of the vehicle may change as it ages, necessitating correction
Statistics based	Song et al.[30]	Analyzed the time-intervals of the CAN messages	Real and synthesized	Offered effectiveness of the method for injection attack detection	Workable only to periodic messages excluding nonperiodic messages
	Young et al.[31]	Analyzed the time-intervals in the frequency domain	Real	Improved frequency-based injection attack detection method	High false-positive rate, and only identify injection attack
ML and DL based	Kang and Kang[32]	Deep belief network	Simulation	Presented unsupervised statistical feature extraction algorithms for ECU messages	Extremely time-consuming in the training phase, evaluation only based on simulation data

(Continues)

TABLE 1 (Continued)

Categories	Research work	Technique	Evaluation data	Contribution	Limitation
	Yang et al.[33]	Signature- and anomaly-based multilayer hybrid IDS	Real	Proposed an IDS that is effective for attacks on both in-vehicle and external networks	Spatial–temporal features and important regional features are ignored
	Tariq et al.[34]	Convolutional LSTM Network and transfer learning	Real	Improved performance of unknown attack detection	Known attacks have poor relative detection performance
	Pawelec et al.[35]	LSTM prediction at the bit level	Real	Avoided reverse engineering proprietary encodings	Limited attack detection range
	Qin et al.[19]	LSTM network and five loss function	Real	Presented two predicted models on different data formats	Relatively poor detection accuracy
	Song et al.[36]	Deep CNN	Real	Reduced inception-resent model complexity to adapt IDS and improve detection performance	The spatial–temporal relationship of CAN frames is ignored
	STC-IDS (ours)	Parallel convolutional LSTM attention network	Real	Important concerns for spatial–temporal features and crucial features	More unknown attacks are ignored

Abbreviations: CAN, controller area network; CNN, convolutional neural network; DL, Deep learning; ECU, electronic control unit; ID, intrusion detection; IDS, intrusion detection system; LSTM, long short-term memory; ML, machine learning; STC, spatial–temporal correlation.

2.1 | ID model based on specification

The specification-based IDS focuses on defining system specifications, such as protocols and frame formats. When packets mismatch the system specification, an exception alarm is raised. In 2016, Dagan and Wool[27] introduced an antispoofing system that detects malicious messages using each ECU, that is, by detecting CAN message ID that was not sent by the ECU itself. Thereafter, the ECU informs the IDS, and then an interrupt pulse is sent to the CAN bus to overwrite the spoofed message. However, each ECU undertakes the IDS role, which increases a certain burden on communication.

In 2018, Studnia et al.[26] presented a signature-based IDS which utilize a list of signature derived from CAN data set. However, this method is subject to the limitation that the length of the CAN bus words may not be known a priori. Recently, Olufowobi et al.[20] proposed a real-time IDS based on specification. The algorithm extracted the timing model to detect anomalies through observing CAN traffic rather than depending on predefined specifications, yet exhibited relatively poor performance in the real attack data set.

2.2 | ID model based on fingerprint

The fingerprint-based approaches are mainly based on profiles defined by ECU characteristics to implement anomaly detection. In 2016, Cho and Shin[28] proposed clock-based IDS to analyze ECUs periodic frequency. The method established the ECUs clock baseline through the recursive least-squares algorithm to detect intrusion. But it is workable only to periodic messages excluding nonperiodic messages.

Interestingly, Choi et al.[29] found a method to establish the electrical signal characteristics of each ECU using the physical layer data of CAN communication, and harness these signal characteristics as the fingerprint for each ECU. Regrettably, the electrical characteristics may change as the vehicle ages, and thus the IDS needs to keep updating.

2.3 | ID model based on statistical

Unlike previous methods, the statistics-based approach implements anomaly detection by means of statistical information obtained from CAN traffic at the network level. Song et al.[30] introduced a lightweight IDS in 2016 that detected anomalies by monitoring the abnormally shortened intervals between messages. Although the proposed algorithm could have highly sensitive to common injection attacks and low computing cost, it cannot detect irregular incoming messages.

Furthermore, Young et al.[31] comprehensively analyzed the frequency characteristics of CAN messages in various driving modes, such as reverse, acceleration, and hold speed, and then proposed a frequency-based IDS. In spite of the high detection accuracy, there is a high false-alarm rate. Manifestly, an IDS based on conditional statistical relationship analysis to learn the normal behavior of the system can detect manipulations and incorrect payload values,[37] but still does not satisfy the high detection rate and low latency required by present-day IVNs for anomalous traffic.[38]

2.4 | ID model based on DL

ML- and DL-based IDSs are an excellent option for extracting and learning normal or abnormal behavior, which provides models with detect and predict ability.[39,40] Kang and Kang[32] constructed a deep confidence network under unsupervised learning to detect if anomalies deviate from normal. However, inefficient and only validation of simulation data is not sufficient.

In 2019, Pawelec et al.[35] proposed a three-layer LSTM neural network to predict the data payload for each CAN ID, which avoided reverse engineering proprietary encoding. Similarly, Qin et al.[19] also implemented anomaly detection for CAN bus based on timing features by LSTM and reconsidered two data formats of CAN frames. Although these methods are implemented at the CAN bits level, they only consider timing characteristics and have relatively poor detection performance. Recently, convolutional neural networks (CNNs) have been implemented for traffic detection and praised for their high detection efficiency.[41]

Song et al.[36] presented a reduced inception residual network to construct an IDS capable of detecting spoofing and denial of service (DoS) attacks in a continuous pattern of vehicular traffic. Since the assistance of spatial–temporal relation is not taken into account, there is still room for improvement in the false-positive rate. In other studies, Tariq et al.[34] introduced a convolutional LSTM-based ID method. Although it displayed excellent detection performance in unknown attacks than transfer learning, known attacks performance relatively worse than may be caused by the relevance of the CAN message features being discarded. Moreover, a multitiered hybrid IDS that incorporates a signature-based IDS and an anomaly-based IDS is proposed to detect both known and unknown attacks on vehicular networks by Yang et al.[33] in 2021. Their model has been proven that is effective for attacks on both in-vehicle and external networks. However, modeling with spatial–temporal features might take performance a step further.

However, these ML- and DL-based methods have different in selecting the detection domain, typically the detection arbitration domain, the detection data domain, and the similar to our work that the spatial–temporal feature extraction. In a nutshell, traditional methods based on specification, fingerprint, and statistical have limitations in terms of reliance on anomaly feature libraries, message frequency, message time domain, and fingerprint information. Instead, it is imperative in the ML and DL area to improve automotive IDS detection performance and reduce false positives by complementing spatial–temporal features in a limited message communication mode. Furthermore, the model trained with limited spatial–temporal features will not effectively improve the detection performance.

Considering the nature of attention mechanisms to capture essential features, the generation of spatial–temporal attention features is one of the potential ways to address the above problem.[42] Hence, modeling the normal behavior of CAN packets in combination with spatial–temporal attention features and then discovering the difference between anomalies and target traffic is still one of the ways to improve in-vehicle IDS.

3 | AN IN-DEPTH OVERVIEW OF CAN BUS DATA SET

3.1 | Vulnerabilities of IVNs

Intelligent connected vehicles (ICVs), integrating modern computing and communication technologies, are designed to improve user experience and driving safety. As the most significant communication medium, the CAN is the most prevalent bus topology network employed in contemporary vehicles owing to its low cost and complexity, high reliability, and fault-tolerance characteristics.[43,44] All ECUs, connected to the CAN bus, are capable of exchanging messages in the form of data frames.[45,46] Figure 1 presents the structure of a CAN message, consisting of seven fields[47]: (1) start of frame (1 bit); (2) arbitration field (12 bits for standard frames and 29 bits for extended frames); (3) control field (6 bits); (4) data field (maximally 8 bytes); (5) cyclic redundancy code (CRC) field; (6) acknowledge (ACK) field; and (7) end of frame.

In the entire CAN frame, the most important is the arbitration field and the data field, as the arbitration field determines the priority of the message,[48] shown in Figure 2A; the data field contains the actual transmitted data that defines the node actions. Moreover, if an error is detected by the CRC field, the receiving node will discard the received error message, while the sending node will only assume a transient fault on the bus and enter arbitration to resend the message frame,[49,50] shown in Figure 2B. To guarantee the system consistency, the ECU broadcasts messages at regular intervals in spite of the data values have not changed. However, security problems were poor-needy thought out at the beginning of the CAN bus design,[51] including its broadcast transmission strategy, lack of authentication and encryption, and unsecured priority scheme. Hence, many network traffic injection attacks are possible.

This directly motivates the adversary to attack IVNs in a variety of ways, as shown in Figure 3. Clearly, the adversary can not only through the On-Board Diagnostics II (OBD-II) port for physical attacks but also implement a remote attack easily (e.g., Wi-Fi or Bluetooth).[10] Types of such attacks include flooding the bus with messages designed to circumvent legitimate

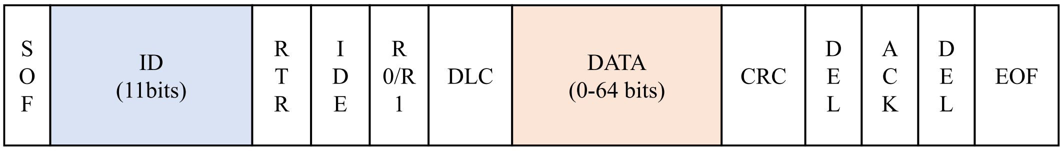

FIGURE 1 Structure of a CAN 2.0A message frame. ACK, acknowledge; CAN, controller area network; CRC, cyclic redundancy code; DEL, delimiter; DLC, data length code; EOF, end of frame; ID, intrusion detection; IDE, identifier extension; RTR, remote transmission request; SOF, start of frame. [Color figure can be viewed at wileyonlinelibrary.com]

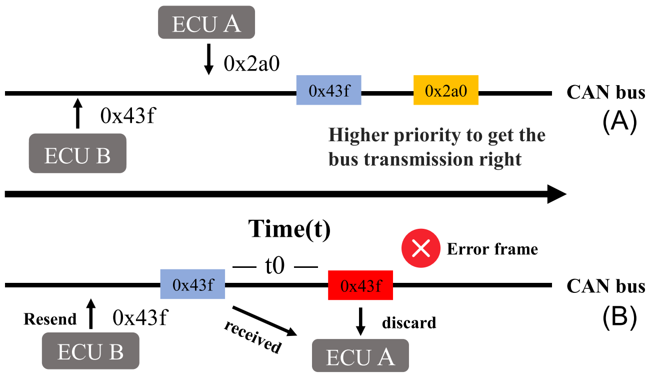

FIGURE 2 Conceptual diagram of the message priority and CRC detection. CAN, controller area network; CRC, cyclic redundancy code; ECU, electronic control unit. [Color figure can be viewed at wileyonlinelibrary.com]

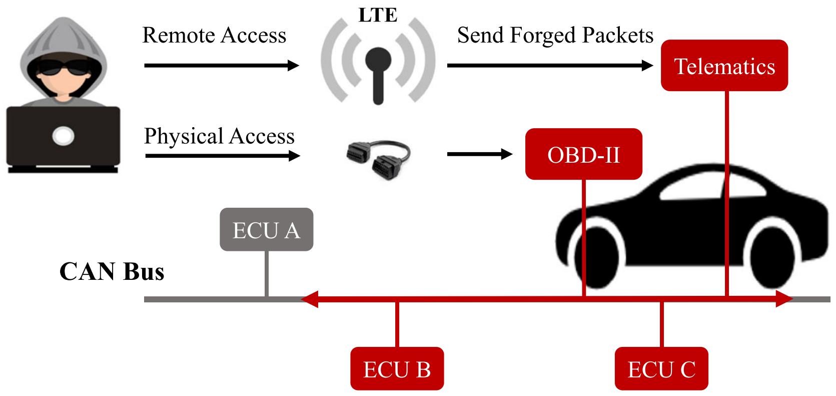

FIGURE 3 Scenario in which an attacker performs an injection attack can be a remote attack or a physical attack can be implemented. CAN, controller area network; ECU, electronic control unit; LTE, Long-Term Evolution; OBD-II, On-Board Diagnostics II. [Color figure can be viewed at wileyonlinelibrary.com]

messages or using spoofed bus identifiers with invalid information.[36] Furthermore, there are more sophisticated and stealthy attacks.[52] These attacks appear to be legitimate traffic sequences that are tough to distinguish from normal messages.[1]

Once the attacker has successfully compromised, it has the opportunity to forge ECU nodes and take control of incumbent nodes to inject nefarious messages.[53,54] In the CAN protocol, the connected nodes are synchronized with the current vehicle state by accepting the data field bits of the frame.[36] Consequently, to successfully deceive the ECU, the adversary must insert tampering messages in a high frequency and priority manner by following the target CAN ID message immediately after the message.[55] If the attacker injects a high-priority CAN frame, where the data field is populated with a status command to turn off the "wiper," the driver loses judgment and even serious traffic accidents in rainy conditions while driving at high speed.[56]

3.2 | STC feature analyzing for CAN data set

In this paper, we utilized car-hacking data set which is published by Song et al.[52] This data set is injected with four attacks, respectively, DoS attack, fuzzy attack, spoofing attack, including revolutions per minute (RPM) and Gear, as illustrated in Figure 4. The detailed injection rules are as follows.

1. *DoS attack*: DoS attacks in the data set are to inject a high priority message with a "0x000" CAN ID every 0.3 ms, with the data field populated with 0.
2. *Fuzzy attack*: Fuzzy attacks in data set are injected every 0.5 ms with CAN messages where the CAN ID and DATA values are forged randomly.
3. *Spoofing attack*: Spoofing attacks in data set are injected messages every 1 ms with a specific CAN ID, for example, related to RPM/Gear.

Table 2 indicates the number of normal and injected messages in each attack data set. To summarize the spatial–temporal details of normal CAN bus traffic during the operation of a real vehicle, we first investigated the communication patterns of different CAN IDs. Since the data set is

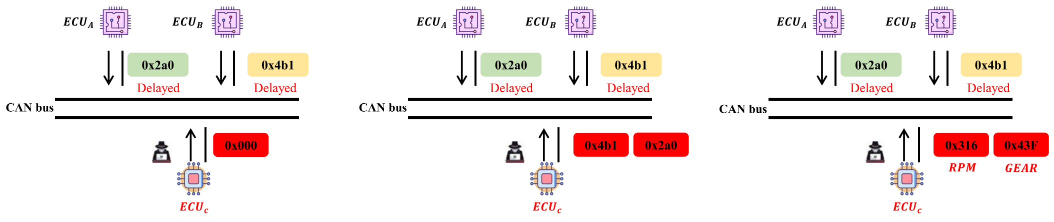

FIGURE 4 Illustration of the injection process of utilized in-vehicle intrusion data set. CAN, controller area network; ECU, electronic control unit. [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 2 Overview of car-hacking data set

Attack type	Normal messages	Injected messages
DoS attack	3,078,250	587,521
Fuzzy attack	3,347,013	491,847
RPM attack	2,766,522	597,252
Gear attack	2,290,185	654,897

Abbreviations: DoS, denial of service; RPM, revolutions per minute.

not publicly available as to the receiving sender and receiver of the data, the paper analyzes the CAN protocol table of a brand from our laboratory. We find that an ECU has a fixed set of CAN IDs (e.g., engine mangement system, containing 0×101, 0×278, and 0×281) and that the recipients of the different CAN IDs are also fixed (e.g., 0×101, containing telematics control unit, electronic stability program, electrical park brake, and T-BOX). This is the initial purpose to design a single-frame detection model capable of tracking unauthorized ECUs and protecting nonattacked ECUs.

Additionally, the frequency of the different IDs is fixed by the vehicle manufacturer. It is worth noting that important automotive components have a higher priority. Hence, time-series features are reflected in the ID of the CAN messages. Despite the fact that there are some event-triggered messages with variable frequency, the set of commands is also fixed. The details are shown in Table 3.

Spatial feature expression, especially the data field of the CAN frame, is most significant. Most new messages are generated at a steady rate over the period of data acquisition in the data set. In this study, the spatial signature analysis was based on the data segment, and a single message often had a regular change in timing. We analyzed this and aggregated the messages of different periodic variation rules. Table 4 displays some CAN message of different CAN IDs from the data set, where the omissions represent the same transferred bytes. We can observe that they have certain fixed byte constants (e.g., the first 7 bytes of ID = 0x260), as well as fields that change more frequently (e.g., the third bytes of ID = 0x316), but only in a certain range. There are distinct spatial characteristics of the data field, useful for modeling the normal behavior of the CAN packet, and also inspire the requirement to design attention convolutional blocks.

To more visually observe the byte change patterns, the data fields for each message are displayed in a heat map, as shown in Figure 5. To apparently visualize the differences in variation of each byte, this paper presents 100 consecutive communication messages based on three important CAN IDs, such as gear and speed. There is a certain period of color shade variation at ID = 0x260 and ID = 0x43F, while the messages sent by the RPM-specific IDs are clearly irregular, corresponding to the summary in Table 4.

TABLE 3 Partial ECU transmission, reception, and time cycle of an automotive brand

ID	Transmitter	Periods	Receiver
0x101	EMS	10	TCU, ESP, EPB, T-BOX
0x278	EMS	10	TCU, GSM, ABS, ESP, EPS, EPB, PEPS
0x281	EMS	100	TCU, AC, ICU, HUD, T-BOX
0x1A0	TCU	10	EMS, GSM, ESP, EPB, PEPS, DRC, PDC
0x211	ESP	10	TCU, ESP, EPB, T-BOX
…	…	…	…
0x2EA	ABS	20	EMS, TCU, EPB, ICU, T-BOX, APA
0x68A	NVS	500	HUD, PSW

Abbreviations: ABS, antilock brake system; ECU, electronic control unit; EMS, engine mangement system; EPB, electrical park brake; ESP, electronic stability program; GSM, global system for mobile communications; HUD, head up display; NVS, night vision system; PEPS, passive entry & push start; PSW, part submission warrant; T-BOX, telematics BOX; TCU, telematics control unit.

TABLE 4 Analyzing range statistics of message frame

ID	Transmitter	1	2	3	4	5	6	7	8
0x260	N/A	05	22	00	30	FF	99	63	38
									0B
									1A
									29
0x316	RPM	05	22	6A	0B	21	18	00	7F
			22	16		22			
			23	3A		23			
			24	1A		24			
0x43F	Gear	10	50	60	FF	46	28	0A	00
							0C		
							10		
							F0		

Abbreviation: RPM, revolutions per minute.

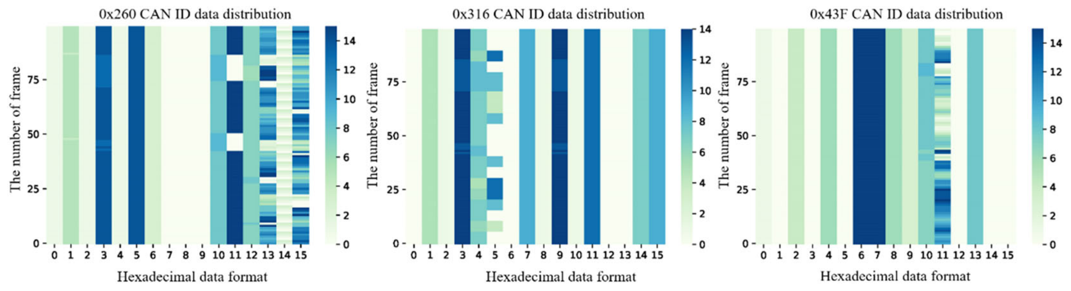

FIGURE 5 Value distribution by heatmap in hexadecimal form of each bit of CAN frame for different ID includes ID = 0x260, ID = 0x316, and ID = 0x43F. CAN, controller area network; ID, intrusion detection. [Color figure can be viewed at wileyonlinelibrary.com]

In conclusion, spatial–temporal details are useful for modeling the normal behavior of CAN packets, and attention also focuses on bytes that change frequently and time-series important relationships, thus helping the model to quickly detect violations and determine the targeted traffic.

4 | IDS USING STC FEATURE

4.1 | Data set preprocessing

It is impractical to train an IDS based on the neural network on the original CAN data set, so data preprocessing is a necessary part before model training. The payload field is 8 bytes as shown in Table 4, where each byte is represented by two numbers in hexadecimal format. In fact, the public source data set was progressively judged and found to contain a large number of data frames that were inferior to 8 bytes or irregular. To ensure uniformity of model input, we filled in the missing data frames with "00" in two scenarios: (1) where the data frame is less than 8 bytes and (2) where only the single digit "0" is used to represent a byte. Afterward, we split and transformed the arbitration bits and data domain in the data set into a trainable data set containing 19 features, respectively, 16 features in 8 bytes of the data domain. In particular, the CAN ID is partitioned into 3 bits to harmonize the operation with the split data field. On the one hand, the combination of hexadecimal features of the original multibit will not be too far removed from the data field features due to the introduction of temporal-frequency features; on the other hand, all messages provided by the public data set have only three valid bits in hexadecimal.

In addition, the values of all features are converted to decimal from hexadecimal. After implementing missing data padding and decimal conversion to meet the basic inputs for the model, several additional data preprocessing steps still need to be completed. First, the CAN frame type is encoded using a label encoder, which is used to convert categorical features into numerical features owing to many ML- and DL-based algorithms that cannot directly support string features.[33] Thereafter, the network data set is normalized by the Min–Max algorithm, as the features collected in network traffic data often have a wide range of differences that impose model deviations and emphasize only large-scale features. Furthermore, the ML- and DL-based models are proven to perform more convergent easily on normalized data set.[57,58] Hence, the data normalization by the Min–Max method is calculated as

$$X_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}. \tag{1}$$

The method implements equal scaling of the original data, where X_{norm} is the normalized data, X is the original data, and $X_{\max}$ and $X_{\min}$ are the maximum and minimum values of the original data set, respectively. Table 5 presents the CAN data that is available as model input, where the former three columns are the CAN ID feature fields, the next eight columns in each message present 16 data domain fields, and the last column represents the label in digital form for each message. In addition, the first two rows represent the distribution for normal CAN data features, while the distribution for injection attacks is the third row. For multiframe detection, the encoder requires an additional image

TABLE 5 Preprocessed data set is represented with the labels divided into normal (0) and intrusion (1)

ID			Data								
ID1	ID2	ID3	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Label
0.2	0.06	0.4	0	0.13	0.4	0	0.13	0.13	0	0.4	0
			0.2	0.06	0.53	0.6	0.06	0.06	0	1	
0.06	0.53	1	1	0.33	0	0	0	0.2	0	0	0
			0.93	0.73	0	0	0	0.8	0	0	
0	0	0	0	0	0	0	0	0	0	0	1
			0	0	0	0	0	0	0	0	

conversion step that splits the collected data set into 64 × 19 two-dimensional (2D) images. After the normalization completing, to prevent the same attacks from appearing in both the test and the segmented attack data, data set division is for 10-fold cross-validation via StratifiedKFold function in sklearn library.

4.2 | STC-IDS model design

The STC-IDS consists of two steps: encoding and detection. The encoder is an analysis of the spatial–temporal characteristics of the CAN messages, and capturing the important relationships based on attention to it; the detector achieves anomaly classification of the valuable spatial–temporal features.

4.2.1 | STC-IDS for single-frame detection

From the perspective of single-frame ID, our aim is to retrieve the illegally controlled source ECU in conjunction with the CAN ID (i.e., accurate identification of every abnormal traffic). In the training phase, we extract spatial features at 1D data by CNN. Since the input of the proposed model is defined as 1 × 19, the spatial component extracts valuable features only through three convolutional blocks and a global pooling layer to avoid invalidating features because of the deeper network. Each convolutional block consists of a 1D convolutional layer, a batch-normalization layer, and an activation function Rectified Linear Unit (ReLU). The batch-normalization allows the model to discard the learning of biases, and make the convolutional output of the model homogeneous particularly. The ReLU function makes the network realize nonlinear feature mapping, while the global pooling serves to assist the model in searching for where critical bytes are and reducing redundant information.

Attention actually mimics a visual mechanism of the human brain, seen as an automatic weighting scheme.[42] Hence, an improved LSTM structure with the attention mechanism (A-LSTM) is designed as a temporal component. We recognize the input as a multivariate time series with a single time step that is initially handled by the dimensional shuffling layer to increase the multivariate processing speed. When the features are fed into the A-LSTM blocks, it can mine significant temporal features. To prevent overfitting, the discard layer plays a crucial role. Overall, the A-LSTM component could discover crucial byte changes, calculated as

$$a_b = \exp\left(u_b^T * u_w\right) / \sum_b \exp\left(u_b^T * u_w\right), \tag{2}$$

where u_w is the weight matrix, and u_b means the implicit representation of the hidden state h_b at computation feature bit b. The u_b is calculated as

$$u_b = \tanh(W_w h_b + b_w), \tag{3}$$

where W_w is the weight matrix and b_w is the bias. Afterward, we attain the attention probability distribution value at each byte. Finally, the final feature vector v is calculated as

$$v = \sum_b a_b * h_b. \tag{4}$$

In Figure 6, we present a parallel feature extraction classification model for single-frame detection. Both features are finally aggregated by the fully connected layers. We refer to it as the CAN valuable feature for STC under multiview learning. Finally, it is fed into the final classification component, which has specific elaboration in algorithm 1.

Algorithm 1 STC-IDS for single-frame

```
1: Require: Input Data $X$, LSTM-Times = 2, Convolutional-Times = 3;
2: Temporal Phase:
3: $X_{\text{temporal}}$ = Dimension shuffle($X$);
4: for Times to LSTM-Times do
5:     $h_b$ = LSTM($X_{\text{temporal}}$);
6:     $u_b = \tanh(W_w h_b + b_w)$;
7:     $a_b = \exp(u_b^T * u_w) / \sum_b \exp(u_b^T * u_w)$;
8:     $v = \sum_t a_b * h_b$;
9:     $X_{\text{temporal}}$ = Dropout($v$);
10: end for
11: Spatial Phase:
12: $X_{\text{spatial}} = X$;
13: for Times to Convolutional-Times do
14:     $X$ = Conv1D($X_{\text{spatial}}$)
15:     $X$ = Batch-Normalization($X$)
16:     $X$ = ReLU($X$)
17:     $X_{\text{spatial}} = X$
18: end for
19: $X_{\text{spatial}}$ = Global-Pooling($X_{\text{spatial}}$)
20: $X_{\text{spatial-temporal}}$ = Concatenate($X_{\text{spatial}}$, $X_{\text{temporal}}$)
21: $X_{\text{spatial-temporal}}$ = Dense($X_{\text{spatial-temporal}}$)
22: $\hat{y}$ = Softmax($X_{\text{spatial-temporal}}$)
```

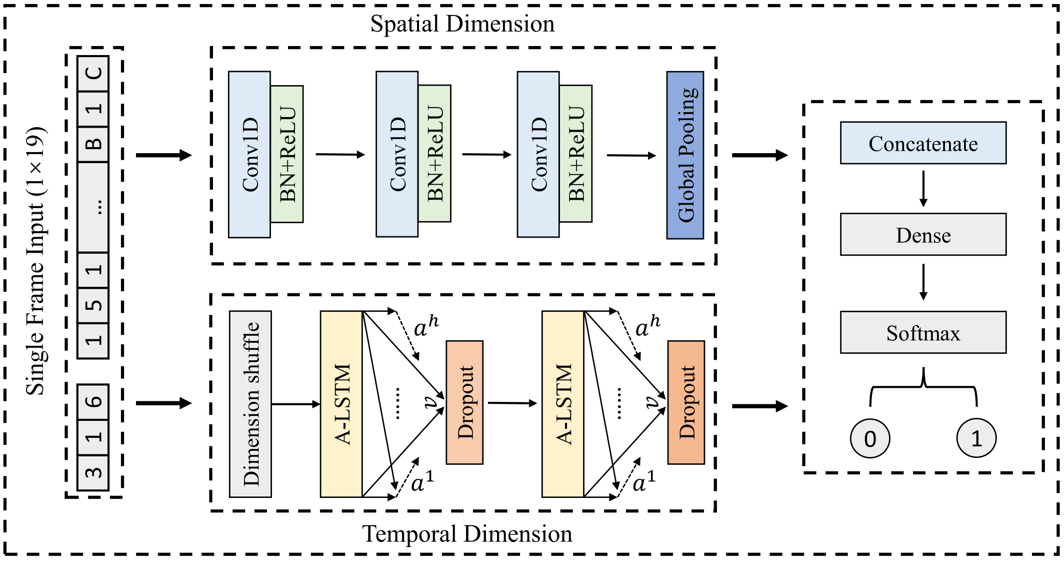

FIGURE 6 To illustrate the single-frame detection neural network. LSTM, long short-term memory; ReLU, Rectified Linear Unit. [Color figure can be viewed at wileyonlinelibrary.com]

4.2.2 | STC-IDS for multiframe detection

To further elevate the efficiency, multiframe-based IDS is constructed. In other words, the model retrieves a continuous CAN 2D matrix, aggregated from 64 consecutive CAN messages during data preprocessing. The matrix height 64 represents the length of the historical time series that the model could recall. For supervised learning, 2D data frames that contain one or more injection messages are marked as attack CAN instance, while data frames that do not contain injection messages are marked as normal CAN instance.

As shown in Figure 7, the 2D data frames are fed into the parallel networks. For the temporal feature extraction component, we remain the structure as same as the single-frame model, but the model input is a time series in 2D so that A-LSTM can pay more attention to important relationships from previous time series. After the dimensional shuffling layer swap out the time dimension of the time series, the processed time series are fed into the A-LSTM blocks. However, the three main parameters in the single-frame temporal attention model, that is, u_b, h_b, a_b, need to be redefined as u_t, h_t, a_t. u_t indicates the implicit representation of the hidden state h_t at computation time t, while a_t represents the final computed attention value.

Moreover, the spatial feature extraction component is inspired by the VGGNet model. We keep the feature extraction capability of the model, but modify the number of convolutional layers and channels to adapt the CAN data set. It is composed of three convolutional blocks, where a convolutional module consists of a 2D convolutional layer, a batch-normalization layer, and a max-pooling layer.

Since the convolutional layer has the feature of shared weight, the proposed model reduces the complexity and improves the inference efficiency. For instance, if a $64 \times 19 \times 3$ feature is mapped into a $62 \times 17 \times 6$ volume, the full-connected layer requires $(64 \times 19 \times 3) \times (62 \times 17 \times 6) = 22M$ weights, while the convolutional layer via 3×3 convolutional kernels only require $(3 \times 3 \times 3) \times 6 = 162$ weights. Figure 8 illustrates the

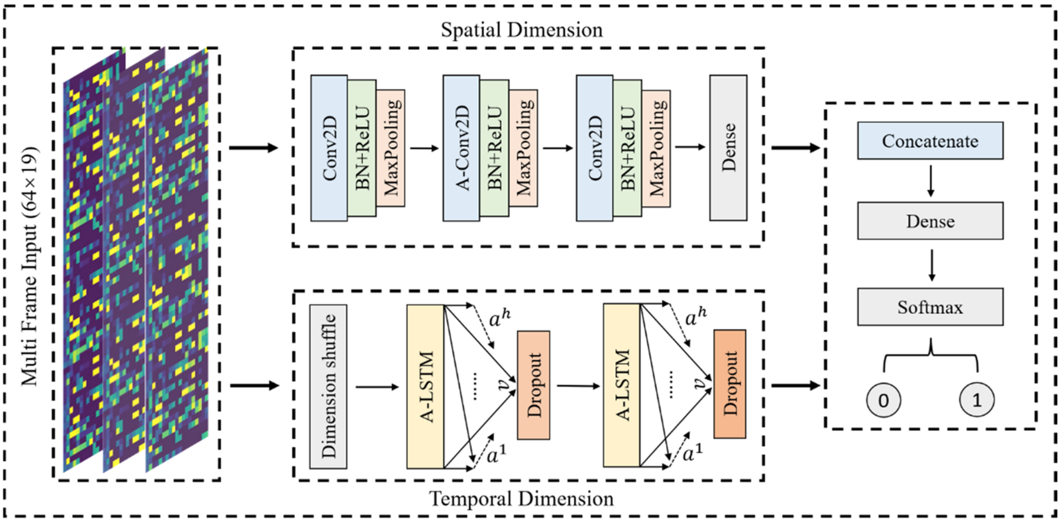

FIGURE 7 To illustrate the multiframe detection neural network. LSTM, long short-term memory; ReLU, Rectified Linear Unit. [Color figure can be viewed at wileyonlinelibrary.com]

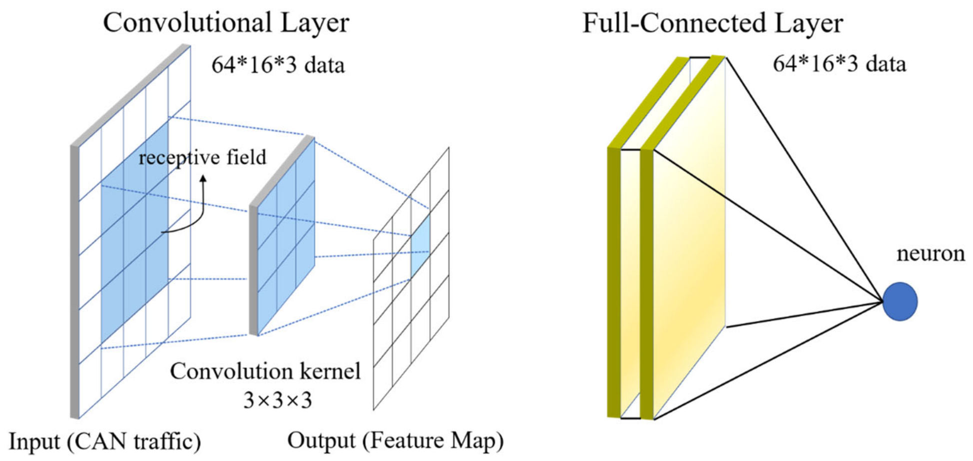

FIGURE 8 In the computation of neurons in the convolutional and fully connected layers, it is clear that the former is smaller than the latter, due to the dense connectivity. CAN, controller area network. [Color figure can be viewed at wileyonlinelibrary.com]

difference between convolutional and fully connected layer computations. The multiframe detection also has the capability to remove redundant information, and reduce the computational effort with the help of the max-pooling layer. The convolutional block maps the raw data to the hidden feature space, thereby performing the task of feature engineering to improve the performance of the spatial component. The fully connected layer serves to map the learned "distributed feature representation" to the sample markup space. Hence, STC features are extracted in a parallel network, and then aggregated by a fully connected layer to finish the classification task.

Most importantly, the second convolutional component is enhanced with visual attention mechanism, called A-Conv2D. The core idea of the component is to help the network extract and represent the information most relevant to the target. For instance, we usually focus on significant information when viewing a photograph, and summarize it. Observing Table 2 and the heat map in Figure 5, there are clearly crucial information changes in the bytes as a 2D data frame. Recently, channel and spatial attention are mentioned in the CNN.[59] Inspired by the idea, the proposed structure integrates such ways, which can assist the model to find where the key information is and where the channel features are learned. The mode has the capability to self-select important features and eliminate the feature engineering step. Hence, convolutional branches of the intermediate layer can obtain a convolutional feature with spatial attention and channel attention. This additional structure is essentially cascaded over the original network with the purpose of better extracting valuable features. In summary, attention convolutional module features are calculated as follows:

$$\begin{aligned} F_{\text{merge}} &= M_s(F_c) \otimes F_c \\ &= M_s(M_c(F) \otimes F) \otimes (M_c(F) \otimes F). \end{aligned} \tag{5}$$

where F_{merge} is the aggregated feature. M_c and M_s are channel attention weight coefficients and spatial attention weight coefficients, respectively, while F is the input feature and F_c is the channel attention feature.

In Figure 9, the feature maps built by the first convolution block are extracted deep features and attention coefficients through the attention convolution component. Initially, the max-pooling and average-pooling layers aggregate spatial attention information to generate two different spatial context descriptions. Immediately afterward, the two features are added together by the multilayer perceptron (MLP). Note that the MLP is weight-sharing to realize

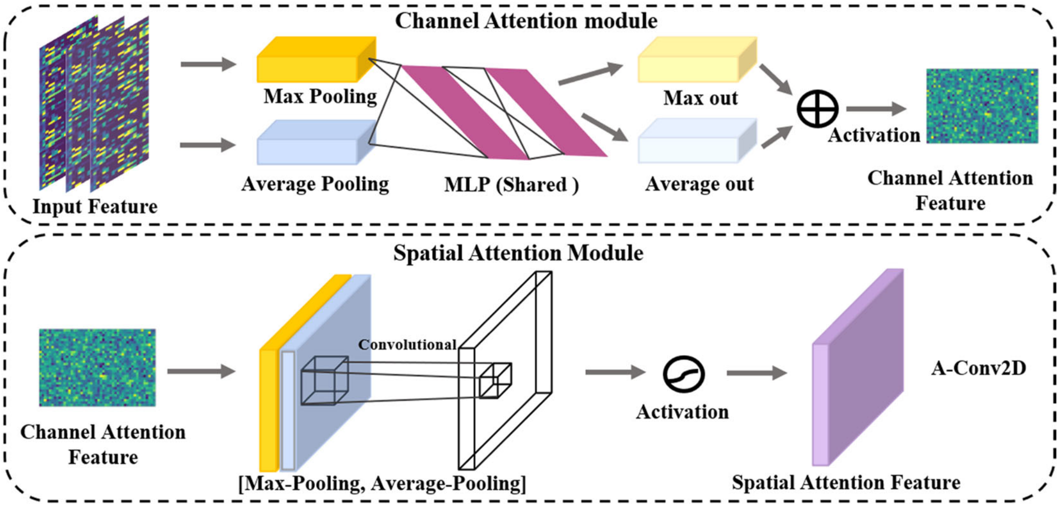

FIGURE 9 Schematic representation of the CAN image computed in the attention convolution module. The channel module utilizes the shared network to output the max-pooling and average-pooling computed features; the spatial attention module pools the pooling output along the channel axis and computes the final features through the convolution layer. 2D, two-dimensional; CAN, controller area network; MLP, multilayer perceptron. [Color figure can be viewed at wileyonlinelibrary.com]

information sharing. Finally, the weight coefficients M_c are obtained through the sigmoid activation function, which is calculated as

$$\begin{aligned} M_c &= \sigma(\mathrm{MLP}(\mathrm{AvgPool}(F)) + \mathrm{MLP}(\mathrm{MaxPool}(F))) \\ &= \sigma\Big(W_1\Big(W_0\Big(F^{c}_{\mathrm{avg}}\Big)\Big) + W_1\Big(W_0\Big(F^{c}_{\mathrm{max}}\Big)\Big)\Big), \end{aligned} \tag{6}$$

After getting the weight coefficients M_c, the channel attention feature F_c is calculated as follows:

$$F_c = M_c(F) \otimes F, \tag{7}$$

where σ is the activation function, and W_0 and W_1 are the weight matrices of the convergence layer.

The spatial attention module is a stable complement to channel attention information because of its ability to mine where the key features are. Similarly, given an $H \times W \times C$ feature F_c, two $H \times W \times 1$ channel descriptions are obtained by averaging pooling and maximum pooling in one channel dimension. Thereafter, the channel descriptions are stitched together based on the channel. Finally, a 2×2 convolutional layer with a sigmoid activation function is applied to obtain the weight coefficient M_s, which is calculated as follows:

$$\begin{aligned} M_s &= \sigma(f^{2\times2}([\mathrm{AvgPool}(F_c); \mathrm{MaxPool}(F_c)])) \\ &= \sigma\Big(f^{2\times2}\Big(\Big[F^{s}_{\mathrm{avg}}; F^{s}_{\mathrm{max}}\Big]\Big)\Big). \end{aligned} \tag{8}$$

where the $f^{2\times2}(\cdot)$ represents the convolutional computation, F^{s}_{avg} and F^{s}_{max} represent the two channel descriptions. Thus, the spatial-channel attention aggregation feature F_{merge} is obtained by multiplying the weight coefficients M_s with the feature F_c. The feature F_{merge} is continued calculated by the normal convolutional component, and then aggregated with the temporal feature to build the spatial–temporal features. The fully connected layer maps the spatial–temporal feature to sample markup space to finish the two-class classification task. Specifically, the multiframe model is described in algorithm 2.

Algorithm 2 STC-IDS for multiframe

```
1: Require: Input Data $X$, LSTM-Times = 2, Convolution-Times = 3;
2: Temporal Phase:
3: $X_{\mathrm{temporal}}$ = Dimension shuffle($X$);
4: **for** Times to LSTM-Times **do**
5:     $h_t = \mathrm{LSTM}(X_{\mathrm{temporal}})$;
6:     $u_t = \tanh(W_w h_t + b_w)$;
7:     $a_t = \exp(u_t^T * u_w)/\sum_t \exp(u_t^T * u_w)$;
8:     $v = \sum_t a_t * h_t$;
9:     $X_{\mathrm{temporal}} = \mathrm{Dropout}(v)$;
10: **end for**
11: Spatial Phase:
12: $X_{\mathrm{spatial}} = X$;
```

(Continues)

```
13: for Times to Convolution-Times do
14:     $F = \text{Conv2D}(X_{\text{spatial}})$
15:     $F = \text{Batch-Normalization}(X)$
16:     $F = \text{ReLU}(F)$
17:     $F = \text{Max-Pooling}(F)$
18:     if Times == 2 then
19:       $M_{\text{c}} = \text{Channel Attention}(F)$
20:       $F_{\text{c}} = M_{\text{c}} \otimes F$
21:       $M_{\text{s}} = \text{Spatial Attention}(F_{\text{c}})$
22:       $F_{\text{merge}} = M_{\text{s}} \otimes F_{\text{c}}$
23:       $X_{\text{spatial}} = F_{\text{merge}}$
24:     else
25:       $X_{\text{spatial}} = F$
26:     end if
27: end for
28: $X_{\text{spatial}} = \text{Dense}(X_{\text{spatial}})$
29: $X_{\text{spatial-temporal}} = \text{Concatenate}(X_{\text{spatial}}, X_{\text{temporal}})$
30: $X_{\text{spatial-temporal}} = \text{Dense}(X_{\text{spatial-temporal}})$
31: $\hat{y} = \text{Softmax}(X_{\text{spatial-temporal}})$
```

4.2.3 | Classification loss

For the ID based on classification, the loss value can be obtained according to the comparison between the prediction label and the actual label. On the basis of this loss message, the loss calculation method is used for the two data formats as discriminant marks for detection. Hence, the predicted values $\hat{y}_i$ are calculated as follows.

$$\begin{aligned}\hat{y}_i &= p(c = i|x)\\ &= \text{softmax}(w \cdot f_x + b) \quad (\text{for i} = 0, 1),\end{aligned} \tag{9}$$

where f_x is the combining features and $\hat{y}_i$ present the probability distribution over target classes zero and one. With the fusion of two structures, the model becomes more complex generating overfit phenomena easily. For better generalization of the model, L_2 regularization is set in the network layer to limit the gradient. Besides, we need to continuously optimize by backpropagation to reduce loss value, and fit the model to the best structure, to maximize the predicted probability p. The loss function is calculated as follows.

$$\begin{aligned}L &= \frac{1}{N}\sum_i L_i + \lambda\|w\|^2\\ &= \frac{1}{N}\sum_i -[y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)] + \lambda\|w\|^2.\end{aligned} \tag{10}$$

5 | EVALUATION

We now validate that the STC feature can be used to detect anomaly frames, and evaluate the performance of a CAN bus prototype and real vehicles.

5.1 | Evaluation metrics and experiment environment

In this paper, statistical metrics true positive (TP) and true negative (TN) are introduced to indicate the number of frames correctly classified as attack and normal, while metrics false positive (FP) and false negative (FN) are introduced to indicate the number of data frames that are misclassified as attack and normal. The model accuracy formula is as follows.

$$\mathrm{Acc} = (\mathrm{TP} + \mathrm{TN})/(\mathrm{TP} + \mathrm{FN} + \mathrm{FP} + \mathrm{TN}). \tag{11}$$

Precision (P) and recall (R) are considered as evaluation metrics to assess classification performance. Precision refers to the rate at which the actual data frame labels are detected correctly, while recall represents the proportion of all attack frame samples that end up in the attack frame class, which are calculated as follows.

$$P = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP}) \tag{12}$$

and

$$R = \mathrm{TP}/(\mathrm{TP} + \mathrm{FN}). \tag{13}$$

The $F1$ score evaluation is also presented, which is a harmonic average based on the detection precision and completeness. Also, the $F1$ score is often used to measure classification performance when the data are unevenly distributed, which is calculated as follows.

$$F1 = 2 \times P \times R/(P + R). \tag{14}$$

Additionally, the false negative rate (FNR) and the error rate (ER) are the one way of assessing classification performance. The FNR is the proportion of frames that are not detected as belonging to the attack frame and the ER is the proportion of frames that are incorrectly classified, calculated as follows.

$$\mathrm{FNR} = \mathrm{FN}/(\mathrm{TP} + \mathrm{FN}). \tag{15}$$

and

$$\mathrm{ER} = (\mathrm{FN} + \mathrm{FP})/(\mathrm{TN} + \mathrm{TP} + \mathrm{FN} + \mathrm{FP}). \tag{16}$$

The two models designed in this paper were trained offline based on the data set, while the testing phase was based on real vehicles, injected with malicious frames of the same rules, to check the performance and efficiency of the models. The following is the experimental training and testing environment.

1. Intel(R) Core (TM) i7-9500U CPU@3.6 GHz.
2. RAM: 64.0 GB.
3. GPU RTX 2080 Ti (training environment).
4. CAN Test, NVIDIA Jetson AGX Xavier (16 GB) (testing environment).

5.2 | Hyperparameter selection and optimization

The selection of hyperparameters is a crucial step in network performance and inference efficiency. Currently, automated hyperparametric optimization (HPO) services and tool-kits address the constant trial-and-error steps of DL developers.[60] In this paper, Bayesian optimization (BO) automatic parametric tuning is used to quickly determine hyperparameters. It is a typical method applied to global optimization problems. Compared with grid search and stochastic search, BO is more computationally efficient and requires fewer attempts to find the optimal set of hyperparameters.[57]

On the basis of the BO optimization library provided by Keras-Tuner and on a 10-fold cross-validation data set, we selected important hyperparameters, such as learning rate (1e − 2, 1e − 3, 1e − 4, 1e − 5, and 1e − 6), optimizer (e.g., Adam, SGD, and RMSprop), dropout rate, and filters. Afterward, we set the optimization goal to validate the accuracy (Val-Acc), a maximum number of trials of 10, and train the model three times per trial. Finally, the optimization results will present the set of the top three hyperparameters for performance.

As shown in Table 6, the average accuracy on the cross-validation set reaches 99.98% in the single-frame model when setting the learning rate is 1e − 6, the dropout rate is 0.4, the filters are 16, 32, and 128, the dense layer is 64, and the optimizer is Adam. Similarly, the accuracy is achieved up to 99.96% to the multiframe model under the optimal hyperparameters.

Interestingly, the inference speed of the model was significantly accelerated aided by the optimal hyperparameters. Despite the higher detection performance was obtained by the single-frame model, the inference speed is slower. Relative to multiframe, it took 50 iterations to converge, whereas the multiframe model only took 30 generations. Figure 10 shows the iterative training losses for each data set. It can be observed that since the DoS attack disrupts the frequency of injection, both models learn the patterns and converge quickly. However, as the complexity of attack data increases, the fuzzy and spoofing attacks injected with random data converge slower in the single-frame model, while the training loss convergence fluctuates significantly in the multiframe model.

TABLE 6 Effect of hyperparameters on model performance under automated HPO

Model type	Learning rate	Dropout rate	Filters	Dense	Optimizer	Val-Acc
Single frame	**1e − 6**	**0.4**	**16, 32, 128**	**64**	**Adam**	**0.9998**
	1e − 6	0.4	8, 96, 32	48	Adam	0.9935
	1e − 6	0.4	8, 16, 192	80	Adam	0.9869
Multiframe	**1e − 2**	**0.4**	**64, 16, 32**	**128**	**Adam**	**0.9996**
	1e − 2	0	8, 16, 32	128	Adam	0.9993
	1e − 2	0	24, 80, 96	256	Adam	0.9992

Note: Best hyperparameters set are highlighted in bold due to the highest validate accuracy.

Abbreviations: HPO, hyperparametric optimization; Val-Acc, validate the accuracy.

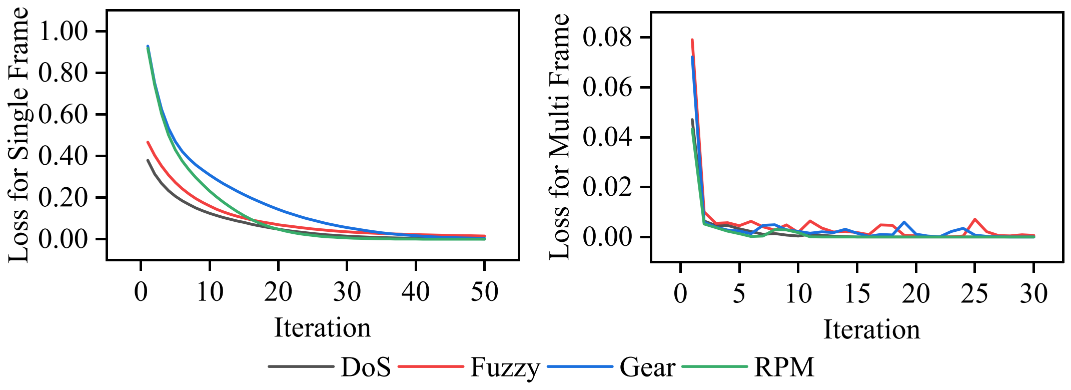

FIGURE 10 To illustrate training loss on each data set in single-frame (A) and multiframe (B) algorithms. DoS, denial of service. DoS, denial of service; ER, error rate; FNR, false negative rate. [Color figure can be viewed at wileyonlinelibrary.com]

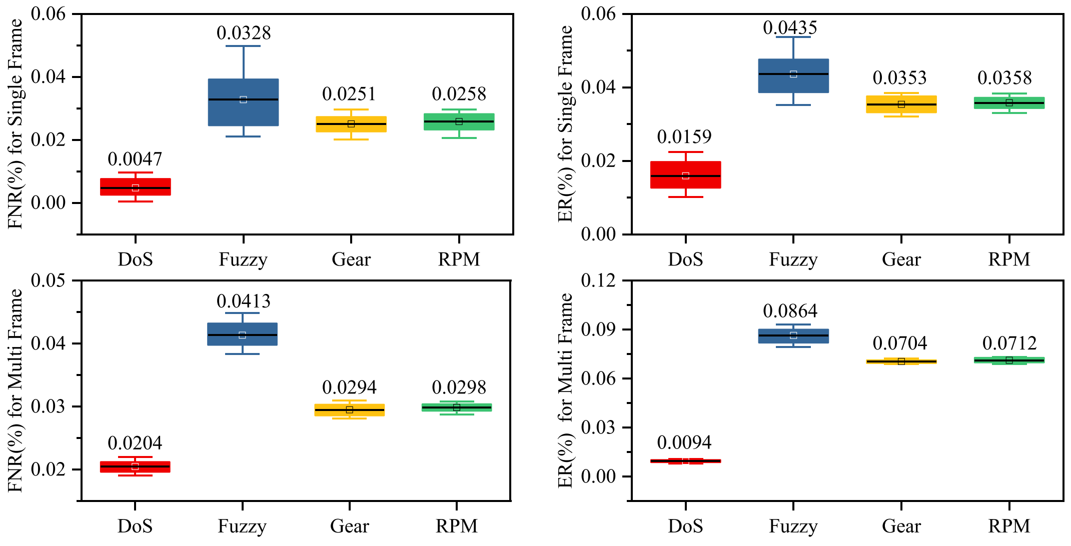

FIGURE 11 Box-plots of single-frame and multiframe models measuring FNR and ER in 30 replicate experiments. DoS, denial of service; ER, error rate; FNR, false negative rate. [Color figure can be viewed at wileyonlinelibrary.com]

5.3 | Detection metrics evaluation and comparison

After training the best two models according to the set hyperparameters, Figure 11 shows the detection performance of the models in terms of ER and FNR for the four-attack testing sets in 30 repeated experiments. The two proposed models, both in terms of FNR and ER, present stable performance to detect DoS attacks with mean values of 0.0047%, 0.0204%, 0.0159%, and 0.0094%. On the contrary, the fuzzy attacks show greater fluctuations in detection performance. The complexity of fuzzy attack data seems to be significantly higher than the other injected data, necessitating more training iterations to construct a stable model. Hence, the ultimate mean values of FNR are still 0.0328% and 0.0413%, as well as ER gets 0.0435% and 0.0864%.

TABLE 7 IDS performance comparison with baseline methods

	ER (%)	FNR (%)	*P* (%)	*R* (%)	*F1* (%)
DoS					
STC-IDS for single frame	0.02	**0.01**	99.95	**99.99**	**99.96**
STC-IDS for multiframe	**0.01**	0.02	99.91	**99.97**	99.94
3-LSTM[35]	0.07	0.22	**1.0**	99.78	99.88
DCNN[36]	0.03	0.10	**1.0**	99.89	99.95
DAE[61]	–	0.12	91.27	99.88	95.36
OTIDS[62]	–	26.2	99.90	73.80	84.88
Fuzzy					
STC-IDS for single frame	**0.05**	**0.04**	**99.97**	**99.95**	**99.96**
STC-IDS for multiframe	0.09	0.07	99.90	99.92	99.91
3-LSTM[35]	0.84	0.65	99.36	99.16	99.26
DCNN[36]	0.18	0.35	99.95	99.65	99.80
DAE[61]	–	3.74	90.05	96.26	93.05
OTIDS[62]	–	29.79	99.14	70.21	82.20
Gear					
STC-IDS for single frame	**0.04**	**0.03**	99.97	**99.96**	**99.97**
STC-IDS for multiframe	0.07	**0.03**	99.94	**99.96**	99.95
3-LSTM[35]	0.24	0.32	99.75	99.68	99.72
DCNN[36]	0.05	0.11	**99.99**	99.89	99.94
DAE[61]	–	18.2	94.63	81.80	87.75
OTIDS[62]	–	28.35	99.83	71.65	83.42
RPM					
STC-IDS for single frame	0.04	**0.03**	99.98	**99.96**	**99.97**
STC-IDS for multiframe	0.07	**0.03**	99.95	**99.96**	99.96
3-LSTM[35]	0.13	0.30	**1**	99.71	99.85
DCNN[36]	**0.03**	0.05	99.99	99.94	99.96
DAE[61]	–	4.27	95.73	95.73	92.10
OTIDS[62]	–	28.32	99.81	71.68	83.43

Abbreviations: DAE, deep autoencoder; DCNN, deep convolutional neural network; DoS, denial of service; ER, error rate; FNR, false negative rate; IDS, intrusion detection system; LSTM, long short-term memory; RPM, revolutions per minute; STC, spatial–temporal correlation.

Although FNR and ER on spoofing attacks are higher than DoS attacks, they achieve stable and better results compared with fuzzy attacks. The single-frame detector averages 0.0251% and 0.0353% for the Gear attack on both metrics, while the multiframe detector obtains 0.0294% and 0.0704%. Similarly, the performance of both models is similar to Gear attacks when detecting RPM attacks. Notably, the single-frame detection model requires mining the characteristics of different CAN packets, thus exhibiting fluctuations in FNR and ER values that are greater than the multiframe model. Despite the multiframe model with better detection efficiency, it has a higher false-alarm rate than the former.

To reflect the advantages of our model, Table 7 lists the detection performance of STC-IDS when compared with those of the other machine-learning techniques, where the highest performance values are highlighted in bold, and "–" only means that the scheme has not been tested on this metric. The results indicate clearly that the STC-IDS model outperforms the previous methods on all data sets. The FNR and ER are significantly reduced. It can be seen that the model captures spatial–temporal details of network traffic remarkably well and enhances the anomaly detection ability satisfactorily.

Evidently, we can observe that the STC-IDS for the single-frame model achieves a stable precision (99.97%), a higher recall (99.96%), and an outstanding *F*1 score (99.96%) on average as compared with a threshold (offset ratio and time interval based intrusion detection system [OTIDS]), classification (deep convolutional neural network [DCNN]), prediction (3-LSTM), and clustering (deep autoencoder [DAE])-based models. In contrast, the performance solely decreases by an average of 0.04% precision, 0.01% recall, and 0.03% in the *F*1 score while maintaining efficiency in the multiframe model.

The 3-LSTM scheme shows high accuracy and unstable recall because of the unconsidered spatial correlation, resulting in a low *F*1 score and a large gap between FNR and ER with our scheme and DCNN. DCNN is currently one of the best models for in-vehicle ID, but the stacking of convolutions is ineffective in capturing the temporal relationships of the inputs. As a result, the single-frame model reduces the FNR and ER by 90% and 33% over the DCNN scheme for DoS attacks, while the multiframe model reduces the FNR by 80% and the ER by 66%. More notably, for complex fuzzy attacks, only 0.04% FNR and 0.05% ER are achieved on the single-frame model. In addition, the FNR is slightly improved on the Gear and RPM data set. Meanwhile, the ER gets approximate performance compared with the DCNN model.

Compared with the DAE model with a lack of labeling constraints, the proposed model has larger improvements in detection performance, while DAE only reaches 95.36% in *F*1 score on DoS attacks, and obtains lower performance on other attacks. Similarly, the OTIDS exhibited a high accuracy rate and extremely low recall, resulting in an FNR of over 25%, reflecting a relatively low *F*1 score. Thus, the robust STC feature demonstrates sufficient benefits in terms of improving detection performance and reducing the false-alarm rate.

5.4 | Time cost in real vehicle

This study first implemented single-frame detection in terms of security considerations. Although the performance is improved over previous algorithms, the efficiency is not guaranteed. Besides, time and resources are the major limitations to applying DL models to real-world vehicle IDS. Therefore, STC-IDS based on multiframes also was proposed. Obviously, significant improvements were made in terms of model convergence time as shown in Section 5.2. To test the efficiency of the proposed model a resource-constrained IVN, the model was tested on an in-vehicle class device the NVIDIA Jetson AGX Xavier. Note that only 4 GB of video memory was allocated for testing. The CAN Test software injected the attack traffic by connecting to the OBD-II port, as shown in Figure 12.

In Figure 13, we present that the proposed model has a lower time (in milliseconds) cost on anomaly detection in the case of different batches compared with previous algorithms. The

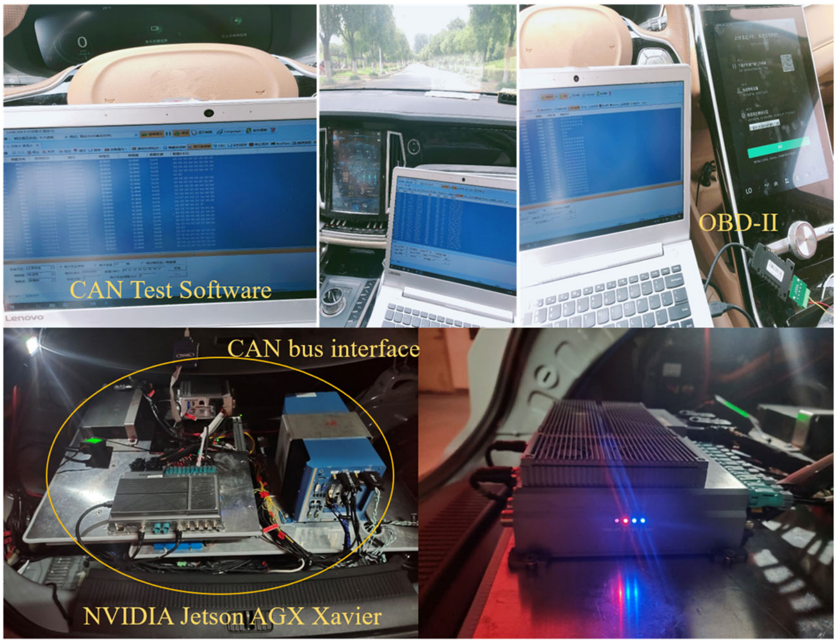

FIGURE 12 To illustrate the testing experiment environment. CAN, controller area network; OBD-II; On-Board Diagnostics II [Color figure can be viewed at wileyonlinelibrary.com]

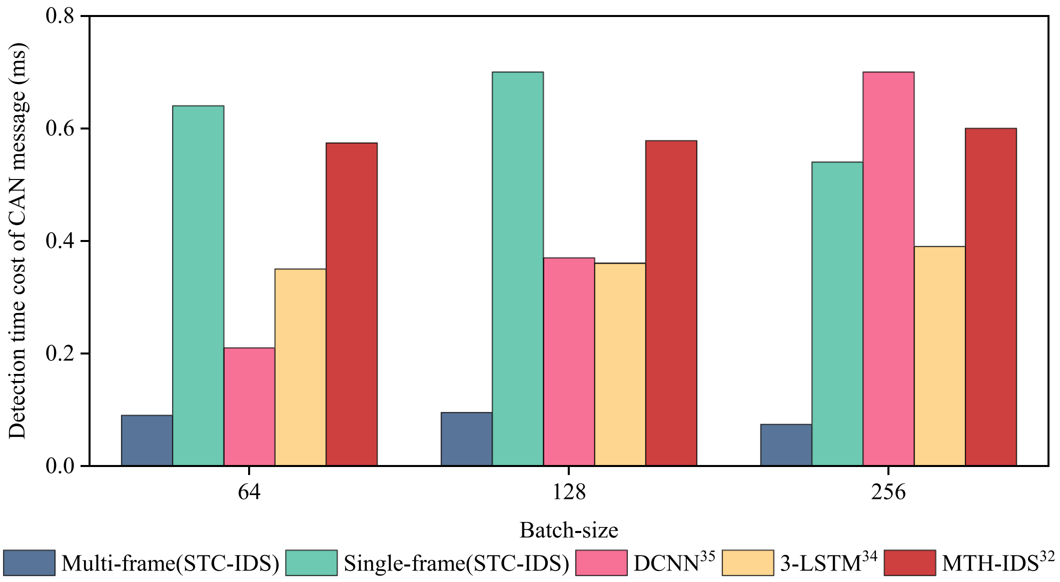

FIGURE 13 Testing time of proposed models on different batches for each message compared with other algorithms. CAN, controller area network; DCNN, deep convolutional neural network; IDS, intrusion detection system; LSTM, long short-term memory; MTH-IDS, multitiered hybrid intrusion detection system; STC, spatial–temporal correlation. [Color figure can be viewed at wileyonlinelibrary.com]

average detection time for the single-frame model remains under 0.7 ms. Although small-batch requires the highest time cost, the detection time is decreased based on the increasingly batch. For example, 256 batch shows 0.54 ms time cost, which is superior to DCNN[36] and multitiered hybrid intrusion detection system (MTH-IDS).[33] Moreover, this truly indicates that the model is constantly learning as it reasoned, thus making it easier to catch malicious features later on. However, it can be estimated that the model can detect 1851 messages in 1 s at the fastest detection time cost, while CAN frames transmit approximately 2000 messages in 1 s. One limitation of this method is that they do not usually satisfy real-time detection.

To overcome this limitation, the multiframe model presents prominent advantages that just need the average time cost of 0.09, 0.09, and 0.074 ms for three batches, respectively. Compared testing time of the previous models as listed in Figure 13, the proposed model not only presents outstanding performance, but also improves about five times in terms of efficiency. This means that the model can infer about five times the number of CAN transmission messages in 1 s. Hence, the proposed model has the feasibility for real-time detection. Unlike the DCNN[36] model, a crucial phenomenon is that the proposed model does not depend on batch size. The same result reflects in the 3-LSTM model[35] and MTH-IDS.[33] But even so, we also suggest a suitable batch size needs to be determined, as large batches may delay anomaly alert.

5.5 | Discussion and limitations

The study presents two enhanced spatial–temporal features analyzing IDS for detecting single CAN messages and consecutive CAN frames based on open injection attack data sets. Experiments indicate that the single-frame model exhibits good detection performance. Similarly, the STC-IDS for multiframe improves efficiency while ensuring performance.

Although the detection efficiency of the single-frame model is right limited, we believe that the model will be suitable for real-time detection when using higher-performance computing devices. In fact, it is necessary to track unauthorized ECUs in conjunction with CAN ID indexing in further in-vehicle security development based on single-frame detection. It is worth noting that the time-cost detection of proposed methods is based on in-vehicle edge computing and autonomous driving platform from our research group. Although it has satisfied the requirement for real-time ID when in a real environment, it has some impact when all tasks are performed simultaneously. Hence, more research still needs to develop in terms of practical implementation.

Clearly, the spatial–temporal feature modeling in this study is based on observing the CAN ID domain, data field, and communication protocol of the specific brand vehicles. The generality consequence of the proposed model is demonstrated due to the fixed time-interval, sender and receiver, despite different vehicle companies making distinct communication protocols. In other words, model transfer only needs to be retrained on the new brand vehicle, which also can extract valuable spatial–temporal features, and obtains excellent detection performance.

Moreover, the division of CAN ID in the scheme is limited to public data sets. To accommodate more realistic in-vehicle messages, the division should be completed in practice by considering both standard frames (11 bits) and extended frames (29 bits). However, it is straightforward only to modify the dimensionality of the input for the proposed model. Most importantly, the model has a fundamental limitation in terms of detecting unlearned types of attacks as it is based on supervised learning. To address this challenge, more research is needed on unknown attack detection using models with generative functions, such as adversarial training or autoencoder.

6 | CONCLUSIONS

This study focuses on learning the temporal and spatial characteristics of IVN traffic to establish enhanced STC features, and then build an automotive ID model. The proposed model is implemented based on the encode-detection architecture. The encoding layer is constructed as a parallel network in both temporal and spatial terms based on LSTM and CNN. The introduction of A-LSTM and A-Conv2D helps the model uncover important relationships between keyword node variation and temporal order. STC features are fed into the detection layer to complete anomaly detection.

Both models achieve optimal hyperparameter selection with BO, reducing the number of iterations and elevating accuracy. Compared with previous schemes, our model achieves a better performance on open source data set, that is, DoS, Fuzzy, Gear, and RPM, especially in the FNR and ER metrics.

Although this study achieves security protection of the CAN bus, there is still much room for improvement, especially unknown attack detection. In future work, we will consider how to express realistic unknown attack messages, improving model robustness and generalization capabilities. Data annotation also is a tedious task, but unsupervised algorithms are one of the solutions, which still need continuous research to improve performance.

AUTHOR CONTRIBUTIONS

Pengzhou Cheng: Methodology, software, validation, data curation, and writing original draft preparation. **Mu Han**: Conceptualization, resource, writing review and editing, supervision, project administration, and funding acquisition. **Aoxue Li**: Supervision. **Fengwei Zhang**: Writing review and editing, and supervision.

ACKNOWLEDGMENTS

This study was supported by the following funds: Natural Science Fund for Colleges and Universities in Jiangsu Province (12KJD580002), the Jiangsu Graduate Innovation Fund (KYLX1057), and the Key Research and Development Plan of Jiangsu Province in 2017 (Industry Foresight and Generic Key Technology; BE2017035).

CONFLICT OF INTEREST

The authors declare no conflict of interest.

DATA AVAILABILITY STATEMENT

Research data are not shared.

ORCID

Pengzhou Cheng http://orcid.org/0000-0003-0945-8790
Mu Han http://orcid.org/0000-0001-7973-8371
Aoxue Li https://orcid.org/0000-0002-5824-1013
Fengwei Zhang https://orcid.org/0000-0003-3365-2526

REFERENCES

1. Song HM, Kim HK. Self-supervised anomaly detection for in-vehicle network using noised pseudo normal data. *IEEE Trans Veh Technol.* 2021;70(2):1098-1108.

2. Kim SH, Seo SH, Kim JH, et al. A gateway system for an automotive system: LIN, CAN, and FlexRay. In: *2008 6th IEEE International Conference on Industrial Informatics*. IEEE; 2008:967-972.
3. Hoppe T, Kiltz S, Dittmann J. Security threats to automotive CAN networks—practical examples and selected short-term countermeasures. *Reliab Eng Syst Saf*. 2011;96(1):11-25.
4. Miller C, Valasek C. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*. 2015;2015(S 91):1-91.
5. Li J, Ye H, Li T, et al. Efficient and secure outsourcing of differentially private data publishing with multiple evaluators. *IEEE Trans Dependable Secure Comput*. 2022;19(1):67-76.
6. Gao C, Li J, Xia S, Choo KKR, Lou W, Dong C. MAS-encryption and its applications in privacy-preserving classifiers. *IEEE Trans Knowl Data Eng*. 2022;34(5):2306-2323.
7. Liu Z, Huang Y, Song X, et al. Eurus: towards an efficient searchable symmetric encryption with size pattern protection. *IEEE Trans Dependable Secure Comput*. 2022;19(3):2023-2037.
8. Checkoway S, McCoy D, Kantor B, et al. Comprehensive experimental analyses of automotive attack surfaces. In: *20th USENIX Security Symposium (USENIX Security)*. Vol 11:2011.
9. Petit J, Shladover SE. Potential cyberattacks on automated vehicles. *IEEE Trans Intell Transp Syst*. 2014;16(2):546-556.
10. Aliwa E, Rana O, Perera C, Burnap P. Cyberattacks and countermeasures for in-vehicle networks. *ACM Comput Surv (CSUR)*. 2021;54(1):1-37.
11. Rouf I, Miller R, Mustafa H, et al. Security and privacy vulnerabilities of {in-car} wireless networks: a tire pressure monitoring system case study. In: *19th USENIX Security Symposium (USENIX Security 10)*; 2010.
12. Yan H, Chen M, Hu L, Jia C. Secure video retrieval using image query on an untrusted cloud. *Appl Soft Comput*. 2020;97:106782.
13. Chen C, Huang T. Camdar-adv: generating adversarial patches on 3D object. *Int J Intell Syst*. 2021;36(3): 1441-1453.
14. Wu W, Li R, Xie G, et al. A survey of intrusion detection for in-vehicle networks. *IEEE Trans Intell Transp Syst*. 2019;21(3):919-933.
15. Tianqing Z, Zhou W, Ye D, Cheng Z, Li J. Resource allocation in IoT edge computing via concurrent federated reinforcement learning. *IEEE Internet Things J*. 2021;9(2):1414-1426.
16. He D, Chan S, Zhang Y, Wu C, Wang B. How effective are the prevailing attack-defense models for cybersecurity anyway? *IEEE Intell Syst*. 2013;29(5):14-21.
17. Cai J, Wang Q, Luo J, Liu Y, Liao L. CapBad: content-agnostic, payload-based anomaly detector for industrial control protocols. *IEEE Internet Things J*. 2022;9(14):12542-12554.
18. Mo K, Tang W, Li J, Yuan X. Attacking deep reinforcement learning with decoupled adversarial policy. *IEEE Trans Dependable Secure Comput*. 2022;18(5):2438-2455.
19. Qin H, Yan M, Ji H. Application of controller area network (CAN) bus anomaly detection based on time series prediction. *Veh Commun*. 2021;27:100291.
20. Olufowobi H, Young C, Zambreno J, Bloom G. Saiducant: specification-based automotive intrusion detection using controller area network (CAN) timing. *IEEE Trans Veh Technol*. 2019;69(2):1484-1494.
21. Tariq S, Lee S, Shin Y, et al. Detecting anomalies in space using multivariate convolutional LSTM with mixtures of probabilistic PCA. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*; 2019:2123-2133.
22. Zhang C, Song D, Chen Y, et al. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In: *Proceedings of the AAAI Conference on Artificial Intelligence*; 2019: 1409-1416.
23. Sun H, Chen M, Weng J, Liu Z, Geng G. Anomaly detection for in-vehicle network using CNN-LSTM with attention mechanism. *IEEE Trans Veh Technol*. 2021;70(10):10880-10893.
24. Kuang X, Zhang M, Li H, et al. DeepWAF: detecting web attacks based on CNN and LSTM models. In: *International Symposium on Cyberspace Safety and Security*. Springer; 2019:121-136.
25. Cai J, Huang Z, Liao L, Luo J, Liu WX. APPM: adaptive parallel processing mechanism for service function chains. *IEEE Trans Network Serv Manage*. 2021;18(2):1540-1555.
26. Studnia I, Alata E, Nicomette V, Kaâniche M, Laarouchi Y. A language-based intrusion detection approach for automotive embedded networks. *Int J Embedded Syst*. 2018;10(1):1-11.
27. Dagan T, Wool A. Parrot, a software-only anti-spoofing defense system for the CAN bus. In: *ESCAR EUROPE*. Vol 34:2016.

28. Cho KT, Shin KG. Fingerprinting electronic control units for vehicle intrusion detection. In: *25th USENIX Security Symposium (USENIX Security 16)*; 2016:911-927.
29. Choi W, Joo K, Jo HJ, Park MC, Lee DH. VoltageIDS: low-level communication characteristics for automotive intrusion detection system. *IEEE Trans Inf Forensics Secur*. 2018;13(8):2114-2129.
30. Song HM, Kim HR, Kim HK. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In: *2016 International Conference on Information Networking (ICOIN)*. IEEE; 2016:63-68.
31. Young C, Zambreno J, Olufowobi H, Bloom G. Survey of automotive controller area network intrusion detection systems. *IEEE Des Test*. 2019;36(6):48-55.
32. Kang MJ, Kang JW. Intrusion detection system using deep neural network for in-vehicle network security. *PLoS ONE*. 2016;11(6):e0155781.
33. Yang L, Moubayed A, Shami A. MTH-IDS: a multitiered hybrid intrusion detection system for Internet of vehicles. *IEEE Internet Things J*. 2021;9(1):616-632.
34. Tariq S, Lee S, Woo SS. CANTransfer: Transfer learning based intrusion detection on a controller area network using convolutional LSTM network. In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*; 2020:1048-1055.
35. Pawelec K, Bridges RA, Combs FL. Towards a CAN IDS based on a neural network data field predictor. In: *Proceedings of the ACM Workshop on Automotive Cybersecurity*; 2019:31-34.
36. Song HM, Woo J, Kim HK. In-vehicle network intrusion detection using deep convolutional neural network. *Veh Commun*. 2020;21:100198.
37. Tomlinson A, Bryans J, Shaikh SA. Towards viable intrusion detection methods for the automotive controller area network. In: *2nd ACM Computer Science in Cars Symposium*; 2018:1-9.
38. Zhang X, Wang Y, Geng G, Yu J. Delay-optimized multicast tree packing in software-defined networks. *IEEE Trans Serv Comput*. 2021:1-14.
39. Ai S, Koe ASV, Huang T. Adversarial perturbation in remote sensing image recognition. *Appl Soft Comput*. 2021;105:107252.
40. Hu L, Yan H, Li L, Pan Z, Liu X, Zhang Z. MHAT: an efficient model-heterogeneous aggregation training scheme for federated learning. *Inf Sci*. 2021;560:493-503.
41. Wang W, Zhu M, Zeng X, Ye X, Sheng Y. Malware traffic classification using convolutional neural network for representation learning. In: *2017 International Conference on Information Networking (ICOIN)*. IEEE; 2017:712-717.
42. Jiang N, Jie W, Li J, Liu X, Jin D. GATrust: A multi-aspect graph attention network model for trust assessment in OSNs. *IEEE Trans Knowl Data Eng*. 2022;4(6);2962-2974.
43. Yang L, Moubayed A, Hamieh I, Shami A. Tree-based intelligent intrusion detection system in internet of vehicles. In: *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE; 2019:1-6.
44. Liu Z, Li J, Lv S, et al. EncodeORE: reducing leakage and preserving practicality in order-revealing encryption. *IEEE Trans Dependable Secure Comput*. 2022;19(3):1579-1591.
45. Jadhav AU, Wagdarikar N. A review: control area network (CAN) based intelligent vehicle system for driver assistance using advanced RISC machines (ARM). In: *2015 International Conference on Pervasive Computing (ICPC)*. IEEE; 2015:1-3.
46. Cai J, Qian K, Luo J, Zhu K. SARM: service function chain active reconfiguration mechanism based on load and demand prediction. *Int J Intell Syst*. 2022;37:6388-6414.
47. Al-Jarrah OY, Maple C, Dianati M, Oxtoby D, Mouzakitis A. Intrusion detection systems for intra-vehicle networks: a review. *IEEE Access*. 2019;7:21266-21289.
48. Davis RI, Burns A, Bril RJ, Lukkien JJ. Controller area network (CAN) schedulability analysis: refuted, revisited and revised. *Real-Time Syst*. 2007;35(3):239-272.
49. Tindell K, Burns A, Wellings AJ. Calculating controller area network (CAN) message response times. *Control Eng Pract*. 1995;3(8):1163-1169.
50. Punnekkat S, Hansson H, Norstrom C. Response time analysis under errors for CAN. In: *Proceedings Sixth IEEE Real-Time Technology and Applications Symposium (RTAS 2000)*. IEEE; 2000:258-265.
51. Barletta VS, Caivano D, Nannavecchia A, Scalera M. Intrusion detection for in-vehicle communication networks: an unsupervised Kohonen SOM approach. *Future Internet*. 2020;12(7):119.

52. Seo E, Song HM, Kim HK. GIDS: GAN based intrusion detection system for in-vehicle network. In: *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE; 2018:1-6.
53. Zhu T, Li J, Hu X, Xiong P, Zhou W. The dynamic privacy-preserving mechanisms for online dynamic social networks. *IEEE Trans Knowl Data Eng*. 2022;34(6):2962-2974.
54. Ai S, Hong S, Zheng X, Wang Y, Liu X. CSRT rumor spreading model based on complex network. *Int J Intell Syst*. 2021;36(5):1903-1913.
55. Cai J, Fu H, Liu Y. Deep reinforcement learning-based multitask hybrid computing offloading for multiaccess edge computing. *Int J Intell Syst*. 2022;2021:1-23.
56. Yan H, Hu L, Xiang X, Liu Z, Yuan X. PPCL: privacy-preserving collaborative learning for mitigating indirect information leakage. *Inf Sci*. 2021;548:423-437.
57. Yang L, Shami A. On hyperparameter optimization of machine learning algorithms: theory and practice. *Neurocomputing*. 2020;415:295-316.
58. Yuan F, Chen S, Liang K, Xu L. *Research on the Coordination Mechanism of Traditional Chinese Medicine Medical Record Data Standardization and Characteristic Protection Under Big Data Environment*. 1st ed. Shandong People's Publishing House; 2021.
59. Woo S, Park J, Lee JY, Kweon IS. CBAM: convolutional block attention module. In: *Proceedings of the European Conference on Computer Vision (ECCV)*; 2018:3-19.
60. Yu T, Zhu H. Hyper-parameter optimization: a review of algorithms and applications. *arXiv preprint arXiv:2003.05689*. 2020.
61. Amarbayasgalan T, Jargalsaikhan B, Ryu KH. Unsupervised novelty detection using deep autoencoders with density based clustering. *Appl Sci*. 2018;8(9):1468.
62. Lee H, Jeong SH, Kim HK. Proceedings of the European conference on computer vision (ECCV): a novel intrusion detection system for in-vehicle network by using remote frame. In: *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. IEEE; 2017:57-5709.

How to cite this article: Cheng P, Han M, Li A, Zhang F. STC-IDS: Spatial–temporal correlation feature analyzing based intrusion detection system for intelligent connected vehicles. *Int J Intell Syst*. 2022;37:9532-9561. doi:10.1002/int.23012