

Research on CAN Bus Injection Attack Detection Method for Internet of Vehicles Based on Deep Learning

Haoyu Liu

College of computer science, Beijing Information Science&Technology University, Beijing, 100036, China

ABSTRACT

With the rapid development of Internet of Vehicles (IoV) technology, the Controller Area Network (CAN) bus, serving as the core communication protocol in vehicles, faces severe injection attack threats due to the lack of built-in security mechanisms. This paper proposes an injection attack detection method that integrates multi-dimensional features with a lightweight spatiotemporal hybrid network. The proposed method first constructs a multi-dimensional feature space and realizes adaptive feature weighted fusion through an attention mechanism. On this basis, a lightweight hybrid architecture is designed, where depthwise separable convolution and model pruning techniques are introduced to reduce computational overhead, and a contrastive learning strategy is adopted to enhance the model's discriminative capability for anomalous samples. Experimental results demonstrate that the proposed method significantly reduces inference latency while maintaining high detection accuracy, providing an effective solution for real-time intrusion detection in resource-constrained in-vehicle environments.

KEYWORDS

IoV security; CAN bus; injection attack detection; Deep learning; Feature fusion

1 Introduction

The rapid advancement of Internet of Vehicles (IoV) technology has transformed vehicles from closed mechanical systems into interconnected intelligent terminals, yet this increased connectivity has exposed critical security vulnerabilities in in-vehicle networks^[1]. The Controller Area Network (CAN) bus, serving as the core communication protocol among Electronic Control Units (ECUs), was designed without authentication or encryption mechanisms. Consequently, once an attacker gains physical or remote access, malicious messages can be injected to manipulate safety-critical functions such as throttle, braking, and steering, directly endangering driving safety. Unlike traditional network injection attacks, CAN injection attacks exploit the inherent "indiscriminate trust" of the underlying protocol, where any well-formed message is unconditionally accepted and executed by all nodes, making detection particularly challenging. Existing detection methods have evolved from rule-based statistical approaches to deep learning models^[2-4]. Early methods relying on information entropy or frequency analysis can only identify flooding attacks with conspicuous abnormal frequencies but prove nearly ineffective against stealthy attacks that mimic normal communication patterns. Deep learning-based models, including CNNs, LSTMs, and their hybrids, have demonstrated superior capability by automatically learning temporal dependencies from CAN traffic^[2-5]. However, three major bottlenecks persist: monotonous feature engineering that relies solely on metadata such as CAN ID or time intervals while neglecting the physical signals embedded in the data field; insufficient generalization to modern vehicle platforms due to training on outdated datasets; and prohibitive computational and storage overhead that conflicts with the stringent resource constraints of real-time in-vehicle systems^[6].

To address these challenges, this paper proposes a CAN bus injection attack detection method that integrates multi-dimensional features with a lightweight spatiotemporal hybrid network. The method simultaneously extracts three categories of features from CAN messages—ID sequence patterns, inter-frame time intervals, and byte-level variations within the data field—thereby constructing a comprehensive multi-dimensional feature space. An attention mechanism is then employed to achieve adaptive weighted fusion of these heterogeneous features, enabling the model to dynamically focus on the most discriminative dimensions depending on the attack scenario. On this basis, a lightweight neural architecture is designed by combining one-dimensional depthwise separable convolutions with a Bidirectional Long Short-Term Memory network, which jointly captures local spatial patterns and long-range temporal dependencies^[7]. Model pruning and quantization techniques are further introduced to drastically reduce the parameter count and inference latency, while a contrastive learning strategy is incorporated during training to enhance the discriminative boundary between normal and anomalous samples^[8]. Experimental results on public datasets as well as real CAN traffic from modern vehicles demonstrate that the proposed method achieves high detection accuracy across various attack types, including stealthy masquerade and slow injection attacks, while maintaining an inference latency below one millisecond on resource-constrained embedded platforms. This work contributes an effective and deployable solution for real-time intrusion detection in automotive environments, bridging the gap between high-performance deep learning models and the practical constraints of in-vehicle deployment.

2 Multi-Dimensional Feature Representation and Fusion Mechanism

2.1 Main

To overcome the limitation of existing methods that rely solely on metadata and thus exhibit insufficient perception of stealthy attacks, the proposed approach comprehensively characterizes CAN communication patterns from three dimensions—protocol semantics, temporal behavior, and physical payload—and introduces contrastive learning at the feature level to enhance the discriminative capability against anomalous samples.

2.2 Feature Extraction and Fusion

A sliding window of $W=64$ consecutive messages with a stride $s=4$ is adopted, and the following three categories of features are extracted from each window:

CAN ID Sequence Pattern Features: The IDs appearing within the window are One-hot encoded to form a high-dimensional sparse vector. Simultaneously, the occurrence frequency of each ID is computed to construct a normalized frequency statistic vector. The former captures whether legitimate IDs are missing or illegitimate IDs have appeared, while the latter captures whether the frequencies of ID occurrences adhere to the normal periodic pattern. When an attacker injects forged IDs or deletes normal messages, both the composition and the frequency distribution of the ID sequence deviate from the normal baseline.

Time Interval Features: The arrival time differences Δt between consecutive messages within the window are extracted, forming a temporal vector of length 63. This vector is standardized per window using Z-score normalization to eliminate the influence of communication rate fluctuations across different periods. During DoS or flooding attacks, the Δt values decrease sharply overall; in contrast, slow injection attacks introduce only minor perturbations at localized positions, which are amplified by the normalized values.

Data Field Byte Variation Features: A CAN message carries up to 8 bytes of physical data. The data fields of all frames within the window are expanded byte-wise to form a 64×8 matrix. For each byte position, the mean and variance across the 64 values are computed to characterize the overall distribution; the first-order differences between adjacent messages are calculated to reflect the jump amplitude of the physical signal; and the Hamming distance between adjacent bytes is computed to measure the degree of bit flips at the binary level. During normal driving, physical signals vary smoothly and continuously, resulting in small difference values and Hamming distances. In contrast, fuzzy attacks or replay attacks introduce abrupt changes that violate physical constraints, causing these statistics to deviate significantly from the normal range.

After the above three categories of features are independently encoded into equal-length vectors, instead of applying fixed-weight concatenation, a channel-wise attention mechanism is introduced for adaptive fusion: each of the three vectors is compressed into a scalar via a fully connected layer, and the attention scores are obtained through Softmax normalization. The final fused feature is the weighted sum of each vector and its corresponding score. This mechanism enables the model to automatically focus on the most discriminative dimension under different attack scenarios—when the data field fluctuates drastically while the ID sequence remains normal, the weight of the payload byte features automatically increases, thereby sensitively detecting masquerade injections.

2.3 Contrastive Learning for Enhanced Discriminative Boundary

To further improve the robustness against unknown attacks and stealthy anomalies, a contrastive learning strategy is introduced during the training phase^[5]. Positive pairs are constructed from two non-overlapping message windows within the same normal driving period, while negative pairs consist of a normal window and a window containing attack frames. The NT-Xent contrastive loss is employed to force the cosine similarity of positive pairs in the feature space to approach 1 and that of negative pairs to approach 0. This strategy compresses the feature manifold of normal behavior into a compact cluster, enabling any anomalous sample that deviates from the normal pattern to be separated earlier and more clearly in terms of distance, thereby effectively mitigating the missed detection problem of traditional methods against unseen attack types.

2.4 Lightweight Spatiotemporal Hybrid Detection Model Design

After extracting and fusing the multi-dimensional features, it is necessary to construct a sequence detector that achieves both high accuracy and low latency, addressing the challenge that complex models cannot be deployed in real time within resource-constrained in-vehicle environments. The overall pipeline of the proposed model is: input layer $\rightarrow$ feature fusion module $\rightarrow$ 1D-CNN spatial encoding $\rightarrow$ BiLSTM temporal modeling $\rightarrow$ temporal attention pooling $\rightarrow$ classification output.

2.5 Lightweight Implementation of Spatial Feature Extraction

Since the input is already a sequence of one-dimensional feature vectors, one-dimensional convolution is employed to extract local spatial patterns. To control computational overhead, all standard convolutions are replaced with depthwise

separable convolutions^[4], which decompose a standard convolution into a per-channel depthwise convolution followed by a 1×1 cross-channel pointwise convolution. The specific configuration is: the first layer with 32 filters, kernel size $k=3$, and stride $s=1$; the second layer with 64 filters, also using depthwise separable convolutions, followed by batch normalization and ReLU activation. This design reduces the number of parameters of the convolutional layers to approximately 1/9 of that of standard convolutions without sacrificing the receptive field, laying the foundation for embedding the model on in-vehicle terminals.

2.6 Temporal Dependency Modeling

Two layers of bidirectional Long Short-Term Memory networks are configured, with 96 hidden units per layer^[2]. The bidirectional structure enables the model to simultaneously capture long-range dependencies from both forward history and backward context. The forward and backward hidden states at each time step are concatenated to obtain temporal features with a dimension of 192. A temporal attention layer is further embedded on top, where a learnable context vector scores each time step. Time steps containing critical information such as the onset frame of an attack are assigned higher weights, and the weighted sum is fed into a fully connected layer with a Softmax classifier for binary decision.

2.7 Model Compression and Deployment Optimization

The initial model has a total of approximately 1.8M parameters, making it difficult to deploy directly on resource-constrained in-vehicle platforms. To this end, after training is completed, L1-norm based structured pruning is applied to trim connections with smaller absolute weight values in the BiLSTM weight matrices and convolutional layer filters, with a pruning rate set to 40%. After pruning, the parameter count is reduced to approximately 0.72M. INT8 quantization is then applied to further reduce the model size. On a Raspberry Pi 4B, the single-frame inference latency is controlled within 0.9 ms, the memory footprint is only 2.7 MB, and the CPU utilization is approximately 18% when continuously monitoring a 500 kbps CAN bus stream, satisfying the real-time and resource constraints of in-vehicle environments^[6].

3 Experimental Design and Results Analysis

3.1 Datasets and Preprocessing

Three datasets are employed, and the principle of strict independence among training, validation, and test sets according to vehicle models is adhered to for training and evaluation.

Car-Hacking Dataset: This dataset contains DoS, Fuzzy, and Spoofing attacks, with approximately 3 million messages in total. 70% of normal driving and attack samples are used to construct the training set, and the remaining 30% serve as the test set.

OTIDS Dataset: This dataset includes normal traffic without attacks and four attack types. The split scheme provided by the dataset is directly adopted.

Tesla Model 3 CAN Dataset: This dataset consists of real driving logs collected from a modern vehicle without any attacks. To construct test samples, Replay and Masquerade attacks are simulated via script injection. Specifically, legitimate IDs and data fields from the past 1000 frames are randomly inserted, but with abnormal timestamps, in order to test the detection capability against stealthy injection attacks.

Preprocessing steps: For all the above data, sliding windows of $W=64$ are extracted with a stride of $s=4$. The window label is set to "normal" only if the window contains no attack frames; otherwise, it is labeled as "attack". The time interval features are robustly standardized using the median and median absolute deviation (MAD) within each window.

3.2 Comparative Methods and Evaluation Metrics

The baseline methods for comparison include: SVM and Random Forest based on frequency statistics, one-dimensional CNN, two-layer LSTM, CNN-LSTM concatenated model, as well as the full model proposed in this paper and the variant without contrastive learning.

Evaluation metrics: Accuracy, precision, recall, F1-score, false positive rate (FPR), and inference latency. Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced. Styles named "Heading 1", "Heading 2", "Heading 3", and "Heading 4" are prescribed.

3.3 Experimental Results and Comparative Analysis

(1) As can be seen from Table 1, the proposed method significantly outperforms traditional machine learning and baseline deep learning models across all metrics. The unpruned model achieves the highest accuracy, but its inference latency reaches 1.85 ms, exceeding the available per-frame time window. After 40% structured pruning, the latency is reduced to 865 μ s, meeting the real-time requirement of <1 ms, while the accuracy drops only slightly by 0.7%, still substantially surpassing other methods. This validates the effectiveness of the lightweight design^[6-7].

Table 1 Performance comparison of different methods on the Car-Hacking Dataset

Method	Metric					
	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)	FPR(%)	Inference Latency(μ s/frame)
SVM	85.2	83.1	82.5	82.8	5.2	125
Random Forest	87.8	86.3	85.9	86.1	4.1	180
1D-CNN	91.5	90.2	89.8	90.0	3.0	215
LSTM	92.1	91.0	90.5	90.7	2.8	450
CNN-LSTM	93.2	92.4	91.8	92.1	2.5	510
Ours-full(before pruning)	96.1	95.8	95.6	95.7	1.6	1850
Ours-full(after pruning)	95.4	95.0	94.8	94.9	1.8	865

(2) For the most stealthy masquerade and slow injection attacks, traditional methods perform poorly because they rely solely on metadata. The proposed method fully exploits the physical constraints of data field bytes and the discriminative boundary enhanced by contrastive learning, maintaining a recall rate above 91% for these attack types, demonstrating a significant robustness advantage.

Table 2 Recall (%) comparison for stealthy attack detection (results on Car-Hacking and Tesla datasets)

Attack Type	Model			
	SVM	LSTM	CNN-LSTM	Ours-full(after pruning)
Masquerade Attack	71.3	82.4	86.2	93.5
Slow Injection Attack	68.9	79.8	84.1	91.2
Replay Attack	73.5	85.1	88.7	94.8
Fuzzy Attack	88.2	93.4	94.5	98.2

(3) The ablation study progressively verifies the gains contributed by each module. After adding the payload byte module, the F1-score increases by 4.1 percentage points; attention mechanism fusion contributes an additional 1.2 percentage points; and contrastive learning further brings a 1.1 percentage point improvement, proving that each component has an independent positive contribution to detection performance.

Table 3 Ablation study (on the Car-Hacking validation set)

Model Configuration	Metric	
	Accuracy(%)	F1-score(%)
ID+time interval features only	89.7	88.5
ID+time interval+payload bytes(without attention)	93.2	92.6
Above+attention fusion	94.5	93.8
Above+contrastive learning training(full model)	95.4	94.9

(4) When models are trained on traditional datasets and tested on modern vehicle datasets, baseline methods such as CNN-LSTM experience an accuracy drop of nearly 10 percentage points, whereas the proposed method drops by only about 3.6 percentage points. This indicates that multi-dimensional feature fusion and contrastive learning can effectively mitigate the data distribution shift problem and provide better cross-vehicle adaptability.

Table 4 Cross-dataset generalization capability

Training Set	Test Set	Ours-full Accuracy(%)	CNN-LSTM Accuracy(%)
Car-Hacking	OTIDS	94.8	92.5
Car-Hacking	Tesla	91.2	83.7
OTIDS	Car-Hacking	92.5	84.1

3.4 Resource Consumption and Deployment Evaluation

Actual runtime data on a Raspberry Pi 4B show that the pruned and quantized model occupies only 2.7 MB of memory. When continuously monitoring a 500 kbps CAN bus stream, the CPU utilization is approximately 18%, without interfering with other vehicle functions. This proves that the proposed lightweight strategy can be practically deployed on extremely resource-constrained in-vehicle computing platforms.

4 Conclusion

4.1 Main Contributions

This paper addresses the three core challenges in CAN bus injection attack detection—monotonous feature utilization, cumbersome models, and poor adaptability to new vehicle models—by proposing a detection method that integrates multi-dimensional features with a lightweight spatiotemporal hybrid network. At the feature representation level, a three-dimensional feature space covering ID sequence patterns, time intervals, and physical payload bytes is constructed, and an attention mechanism is introduced to dynamically allocate feature weights. Experiments show that the payload features and attention fusion alone cumulatively improve the F1-score by over 5 percentage points. At the model design level, a hybrid architecture combining depthwise separable convolution and BiLSTM is adopted^[7]. Through structured pruning and INT8 quantization, the parameter count is compressed from 1.8M to 0.72M, and the inference latency is reduced from 1850 μ s to 865 μ s, achieving real-time detection below 1 ms on a Raspberry Pi 4B and verifying the feasibility of in-vehicle deployment^[6]. In terms of robustness enhancement, the introduction of a contrastive learning strategy^[8] enables the model to achieve recall rates exceeding 91% for highly stealthy attacks such as masquerade and slow injection attacks. In terms of generalization verification, in cross-dataset testing, the accuracy of the proposed method drops by only 3.6 percentage points, far superior to the nearly 10-percentage-point drop of the baseline model, demonstrating good cross-vehicle adaptability^[1,5].

4.2 Research Limitations

The current method still has several shortcomings. First, there is a lack of explicit modeling for logical attacks that highly mimic physical continuity; if an attacker forges messages that conform to normal vehicle logic, the detector may miss them. Second, contrastive learning^[8] relies on labeled attack samples to construct negative pairs, making it difficult to cover zero-day attacks and other attack variants. Third, the experiments are conducted on a Raspberry Pi to simulate deployment, and real-vehicle verification on actual gateway ECUs or domain controllers has not yet been performed; the attack data are also generated through offline injection. In addition, the method is primarily aimed at the classic CAN 2.0 protocol and does not cover next-generation communication protocols such as CAN FD and Automotive Ethernet.

4.3 Future Outlook

Future research will focus on the following directions: First, explore unsupervised anomaly detection paradigms based on autoencoders or generative models to eliminate dependence on attack labels and enhance immunity to a wider variety of attacks. Second, introduce incremental learning and model continuous update mechanisms to adapt the detector to communication pattern drift during the vehicle lifecycle. Third, construct a vehicle-cloud collaborative security architecture, where the cloud is responsible for training complex models and delivering updates via OTA, and the vehicle side performs real-time monitoring and uploads suspicious traffic, forming a closed-loop evolution of detection capability. Fourth, extend the detection scope to CAN FD and Automotive Ethernet, and investigate a lightweight Transformer-based cross-protocol joint detection framework.

References

- [1] Zhang, Y.Q., Wang, X.F., Liu, X.F., et al. (2020) A Survey on Internet of Vehicles Security. *Journal of Computer Research and Development*, 57(1): 1-20.
- [2] Kang, M.J., Kang, J.W. (2016) Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security. *PLOS ONE*, 11(6): e0155781.
- [3] Song, H. M., Woo, J., Kim, H. K. (2020) In-Vehicle Network Intrusion Detection Using Deep Convolutional Neural Network. *Vehicular Communications*, 21: 100198.
- [4] Song, H.M., Kim, H.K. (2021) Self-Supervised Anomaly Detection for In-Vehicle Network Using Noised Pseudo Normal Data. *IEEE Transactions on Vehicular Technology*, 70(2): 1098-1108.
- [5] Chen, W., Yang, T., Wu, L.B. (2021) CAN Bus Anomaly Detection Method Based on Improved LSTM. *Acta Electronica Sinica*, 49(5): 984-992.
- [6] Liu, Z.Y., Li, F., Chen, Z. (2022) A Lightweight In-Vehicle Network Intrusion Detection System for Resource-Constrained Environments. *Journal of Software*, 33(8): 2892-2908.
- [7] Howard, A.G., Zhu, M., Chen, B., et al. (2017) MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint*, arXiv:1704.04861.
- [8] Chen, T., Kornblith, S., Norouzi, M., et al. (2020) A Simple Framework for Contrastive Learning of Visual Representations. In: *Proceedings of the International Conference on Machine Learning*. Virtual. pp. 1597-1607.