IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 3, June 2026, pp. 2618~2632
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i3.pp2618-2632

DriveShield: attention-based hybrid neural network for intrusion detection in automotive controller area networks

Vismaya Kootayi Kunnacheri[1], Arul Leena Rose Peter Joseph[2]
[1]Department of Computer Science, Faculty of Science and Humanities, SRM Institute of Science and Technology, Chennai, India
[2]Department of Computer Applications, Faculty of Science and Humanities, SRM Institute of Science and Technology, Chennai, India

Article Info

Article history:

Received Dec 17, 2024
Revised Feb 12, 2026
Accepted Apr 20, 2026

Keywords:

Controller area network
Deep learning
Electronic control unit
Intrusion detection system
In-vehicle network

ABSTRACT

Vehicle network security is important as increasing amounts of connected technology are being added to vehicles nowadays, putting them at risk of cyberattacks. This paper presents DriveShield, a novel real-time intrusion detection system (IDS) that is the first to combine gated recurrent units (GRU), convolutional neural networks (CNN), and long short-term memory (LSTM) with an attention mechanism. The systematic pre-processing pipeline, which includes feature engineering, the synthetic minority oversampling technique (SMOTE) for class balancing, and normalization. The model was validated on the open training intrusion detection system (OTIDS) dataset and the Hacking and Countermeasure Research Lab (HCRL) car hacking dataset. In the HCRL dataset, the model had an accuracy of 96.30% with F1-scores as high as 96% for all kinds of attacks. On the OTIDS dataset, it performed very well in terms of generalization, with a highest accuracy of 99.78% and a weighted F1-score of 99.78%. The addition of an attention mechanism enabled the model to concentrate on the most significant features, providing better adaptability to changing threats. These findings demonstrate the efficacy, scalability, and reliability of the system for in-vehicle network security. The future research will focus on performance on lower-frequency attacks through the study of unsupervised learning methods and real-world deployment trials.

Corresponding Author:

Arul Leena Rose Peter Joseph
Department of Computer Applications, Faculty of Science and Humanities
SRM Institute of Science and Technology
Kattankulathur, Chennai, India
Email: leena.rose527@gmail.com

1. INTRODUCTION

The significance of cybersecurity in these innovative technologies has grown as the automotive industry progresses with connected and driverless vehicles. The increasing integration of contemporary automotive technologies, such as electronic control units (ECU) and controller area network (CAN) bus, has made automobiles vulnerable to a number of cybersecurity issues [1]. In-vehicle networks are computationally supported by the ECU, which regulates a number of subsystems, including entertainment, engine management, and braking; because of its electronic components and networked systems, ECUs have become a prominent target for attackers [2]. Being devoid of encryption and authentication support, the conventional CAN bus protocols are vulnerable to malicious interference and unauthorized access; with greater connectivity, there is greater exposure to cyberattacks, which in turn result in risk of injury to drivers and occupants or compromise privacy or integrity [3]. The CAN bus is a message broadcasting bus that can transmit up to 64 bits of data in frames, as shown in Figure 1, and CAN identifies the physical layer and data

link of open system interconnection (OSI) and provides a simple network solution for quick in-vehicle communication [4]. Its goal was to reduce wiring costs by allowing independent electronic control modules (ECMs) to communicate using single wire pairs, improving driving comfort and safety via smooth input and output coordination [5]; however, security-wise it lacks integrated authentication and encryption, employs a broadcast technique that makes messages accessible to all nodes, and due to its lack of an access control mechanism CAN is susceptible to spoofing and denial-of-service (DoS) attacks [6]. A typical CAN frame has an identification of 11 bits [7], [8] (and an expanded CAN frame has a 29-bit identifier), and because any nodes linked to the CAN bus will receive messages issued by any other node, it is simple for an attacker to join network traffic and read data frames; furthermore, messages transferred between ECUs in a CAN network lack authenticator information, so an attacker can stop a frame from getting to the person it's meant for [9], [10]. Figure 2 illustrates a typical in-vehicle CAN network and the common attack surfaces considered in this study.

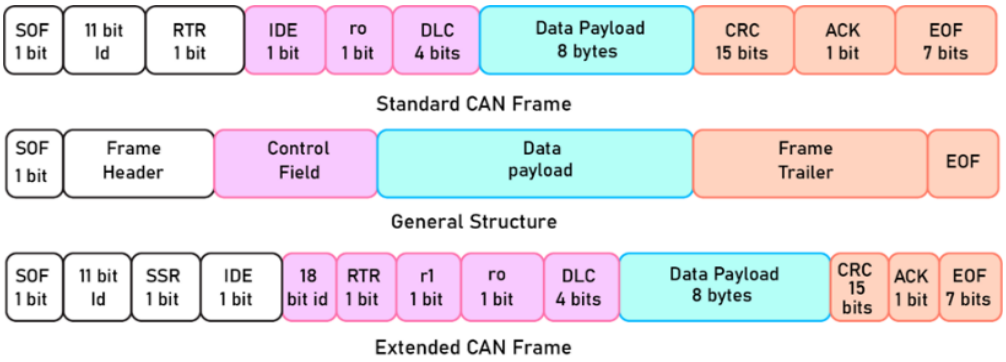


Figure 1. Standard and extended CAN frame formats

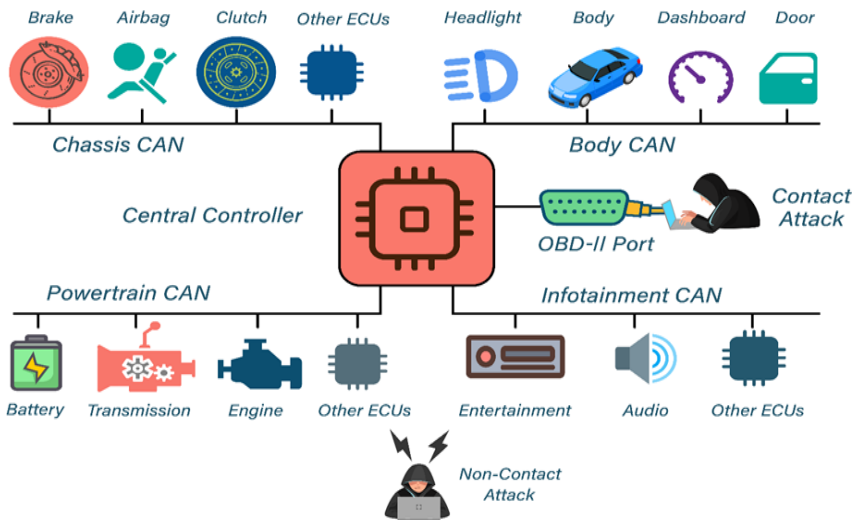


Figure 2. Typical in-vehicle CAN architecture with key ECUs and major attack vectors, including OBD-II contact attacks and remote non-contact attacks

Recent studies have evaluated machine learning and deep learning (DL) methods for CAN intrusion detection, showing promising comparative results for data-driven approaches [11], as well as demonstrating that hybrid deep architectures can significantly improve detection accuracy in real-world in-vehicle environments [12]. With promising detection by DL models, the ability to identify emerging threats, and the emergence of DL, the in-vehicle intrusion detection system (IDS) paradigm takes on a new dimension [13]. Federated and meta-learning approaches have also been explored to enhance personalization and zero-day adaptability in IoT and automotive IDSs [14], reinforcing the need for IDS designs that remain effective under evolving attack strategies and operational constraints.

In spite of growing interest in DL-based IDS solutions for CAN bus networks, most work is based on single-model architectures, which prevents them from simultaneously recognizing both spatial and temporal patterns. For example, Song *et al*. [15] developed an entirely convolutional neural network (CNN)-based IDS optimized for spatial feature extraction and Hossain *et al*. [16] introduced an long short-term memory (LSTM)-only model that was solely concerned with sequential dependencies in CAN messages. Likewise, a lightweight gated recurrent unit (GRU)-based system was proposed to reduce latency in real-time detection [17], yet lacks spatial context integration. Such single-architecture methods tend to fail to generalize over different types of attacks and datasets, and can fail to capture crucial cross-feature dependencies in CAN traffic. Table 1 presents a comparative summary of the existing literature, highlighting key aspects such as model architecture, datasets, detection performance, false positive rates, and class imbalance handling. A visual taxonomy summarizing the evolution of IDS models for in-vehicle CAN networks is provided in Figure 3. Representative examples include a VGG-16-based CNN model by Ahmed *et al.* [18] (high accuracy for DoS and fuzzy with 0.6% false positive rate (FPR), but missing temporal modeling and real-world application), a CNN–LSTM model by Alferaidi *et al.* [19] (up to 99.7% accuracy but without class imbalance handling or temporal attention), and an LSTM-GRU model by Alkhatib *et al.* [20] (high AUC on synthetic traffic but without real-time verification and real CAN traffic generalizability). Related efforts include IDS-IVN by Alqahtani and Kumar [21] (autoencoder-based CNN–LSTM with 99% accuracy and 0.32% FPR, but without deployment feasibility and scalability insights), an MLP-based IDS by Anzer and Elhadef [22] (good classification but shallow temporal processing), and LSTM autoencoders by Ashraf *et al.* [23] (more than 98% accuracy, but less real-time relevance). In addition, Li *et al*. [24] proposed an adaptive deep learning (ADL) model for smart-grid traffic (network security laboratory-knowledge discovery in databases (NSL-KDD)) without direct CAN use or temporal feature modeling, El-Gayar *et al*. [25] proposed DFSENet (98–99% accuracy via parallel ensemble learners but without explicit temporal sequence learning), and Kang and Kang [26] introduced an early DNN-based IDS with deep belief network (DBN) initialization for CAN packet classification, but without current hybrid feature extraction methods and thorough evaluation metrics. A CAN frame contains multiple fields, including the identifier (11-bit standard or 29-bit extended), control field (6 bits, including the 4-bit data length code), data field (0–8 bytes), cyclic redundancy check (CRC) field (15 bits), acknowledgment (ACK) slot, and end-of-frame bits. The broadcast nature of CAN means that all nodes receive every transmitted frame, and the protocol does not provide built-in authentication, which makes injection and spoofing attacks feasible.

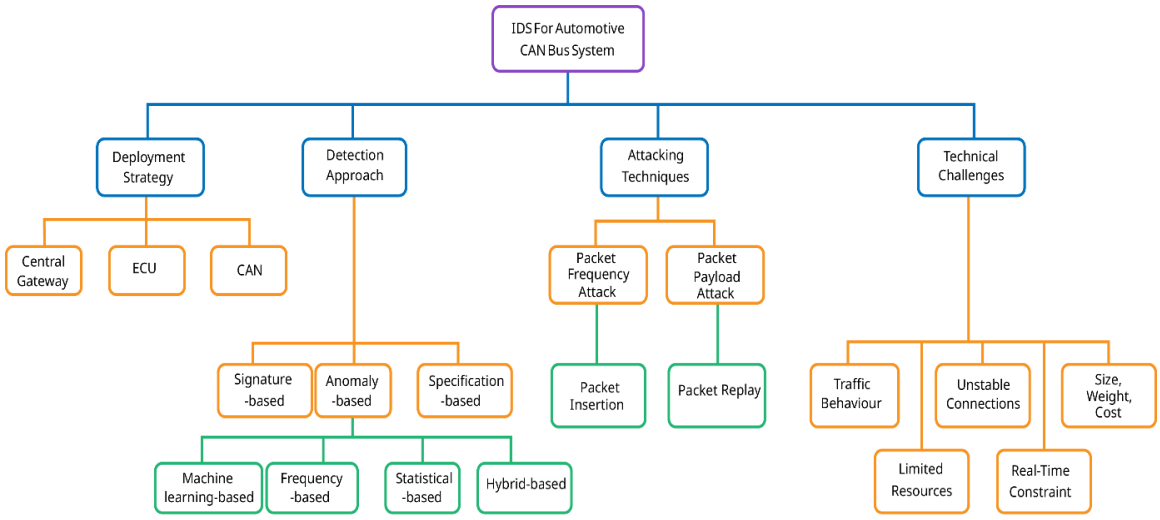


Figure 3. Taxonomy of IDS models for automotive CAN networks

In order to overcome these drawbacks, this paper introduces DriveShield, a DL-based hybrid IDS that combines CNN, LSTM, and GRU layers with an added attention mechanism to dynamically weight important features in spatial–temporal CAN data. This enables the system to prioritize the most discriminative features across time and feature space, improving detection accuracy under diverse attack conditions. Additionally, we test the model on two different, publicly available datasets (Hacking and

Countermeasure Research Lab (HCRL) and open training intrusion detection system (OTIDS) to verify robustness, and add synthetic minority oversampling technique (SMOTE) balancing addressing a major limitation of existing CAN datasets that are often skewed toward normal traffic, multi-metric evaluation, and deployment-ready performance using TensorFlow/Keras in a GPU-empowered cloud environment to show real-time effectiveness. Compared to the studies in Table 1, DriveShield presents a novel hybrid DL model consisting of CNN, LSTM, and GRU structures reinforced by an attention mechanism to efficiently capture CAN data spatially as well as temporally; whereas most previous studies compare on just one dataset or do not address class imbalance, our system is compared on two public CAN datasets (HCRL and OTIDS) with SMOTE-based balancing and performs robustly with as much as 96.30% accuracy and 0.96 F1-score on HCRL, and with 99.78% accuracy and 0.99 weighted F1-score on OTIDS. Also, the model has real-time deployment capabilities and provides extensive multi-metric assessments such as confusion matrices, receiver operating characteristic (ROC) curves, per-class analysis and statistical analysis which makes it a more holistic and generalizable solution for contemporary vehicular networks; to the best of our knowledge, no prior CAN IDS work jointly integrates CNN, LSTM, GRU, and attention, validated across two real-world datasets with statistical significance analysis and deployment-oriented latency benchmarking.

Table 1. Literature review

Study	Dataset	Model	Detection performance	FPR	Class imbalance handling	Computational overhead	Deployment/ resource considerations	Zero-day/ unsupervised capability
Ahmed *et al.* [18]	Simulated CAN (byte-image representation)	CNN (VGG-16)	DoS/fuzzy ≈96%	0.6%	Not addressed	High (image-based CNN)	GPU-oriented image pipeline; high memory footprint	No (supervised)
Alferaidi *et al.* [19]	NSL-KDD	CNN + LSTM	≈ 99.7%	Low (not reported)	Not addressed	High (deep CNN-RNN)	Distributed Spark-based training; not ECU-oriented	No (supervised)
Alkhatib *et al.* [20]	Synthetic SOME/IP	LSTM + GRU	AUC >0.8	Not specified	Not specified	Moderate (recurrent model)	Synthetic traffic only; no embedded deployment analysis	Limited
Alqahtani and Kumar [21]	ROAD (CAN)	CNN + LSTM Autoencoder	≈99%	0.32%	Not handled	Moderate–High (autoencoder)	Reconstruction-based inference increases the runtime cost	Partial (anomaly-based)
Anzer & Elhadef [22]	KDD-99	MLP	High (not specified)	Not reported	Not discussed	Low (shallow network)	Lightweight model; lacks temporal modeling	No
Ashraf *et al.* [23]	Car-Hacking and UNSW-NB15	LSTM Autoencoder	Car >99%, UNSW ≈98%	Not reported	Not handled	Moderate	Offline evaluation; no latency profiling	Partial (anomaly-based)
Li *et al.* [24]	NSL-KDD (Smart Grid)	Adaptive DL + ML	Improved over ML baselines	Not reported	Not explicitly addressed	Moderate	Non-CAN domain; grid-oriented deployment	Limited
El-Gayar *et al.* [25]	CICIDS and Car-Hacking	DFSENet (Ensemble ML)	CICIDS ≈99.2%, Car ≈98%	≈4.4%	Data balancing applied	High (ensemble inference)	Increased memory and inference cost	No
Kang and Kang [26]	In-vehicle CAN	DNN (DBN-initialized)	Improved detection ratio	Not reported	Not handled	Moderate	Early ECU-oriented DL; shallow architecture	No
Proposed Work (Drive Shield)	HCRL and OTIDS	CNN + LSTM + GRU + Attention	HCRL =96.30%, OTIDS =99.78%	Low (≈0.01%)	SMOTE (training only)	Moderate (hybrid model)	Baseline FP32 inference latency measured; GPU-based evaluation; ECU-oriented optimization roadmap	Supervised (extensions discussed)

2. PROPOSED APPROACH

In the next section, we provide the dataset description, data preprocessing, and model architecture. This section explains how the raw CAN traffic is prepared and transformed into model-ready input sequences. Figure 4 shows the overall design and workflow of the proposed method.

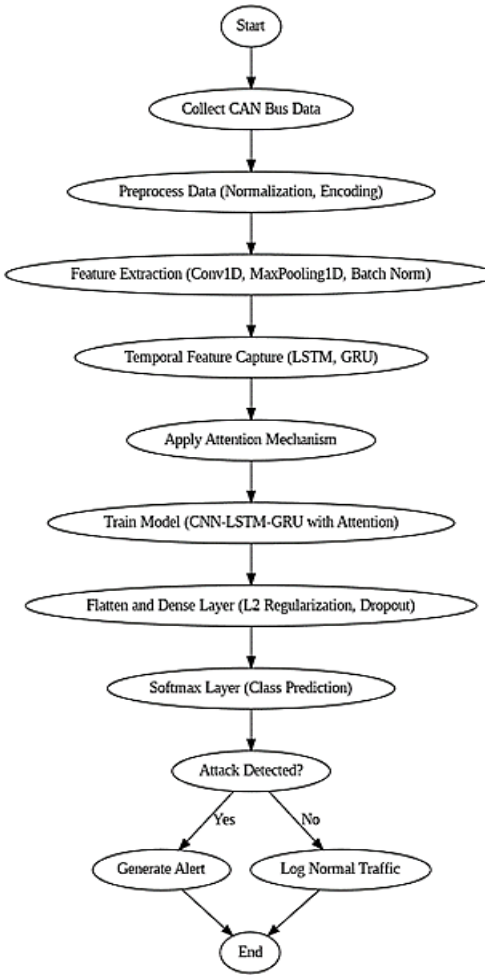


Figure 4. Shows the proposed model and experimental methods applied

2.1. Dataset description

The HCRL CAR hacking dataset [27] and OTIDS dataset was utilized to assess the suggested approach. HCRL comprises actual CAN traffic captured through the OBD-II port under the message injection attacks, such as DoS, gear spoofing, revolutions per minute (RPM) spoofing, and fuzzy attacks. Attacks include injecting CAN messages with high frequency (e.g., every 0.3–1 ms), where each dataset includes approximately 300 injection events over 30–40 minutes of traffic. Each attack lasts for 3–5 seconds. The data includes columns like CAN ID, timestamp, DLC, DATA [0–7], and flag.

The OTIDS (CAN intrusion detection) dataset [28] was gathered from a genuine KIA SOUL vehicle through the OBD-II port under normal drives and controlled attacks. It comprises four types: benign traffic, DoS attacks, fuzzy attacks, and impersonation attacks. DoS consists of high-frequency injection of CAN ID '0×000'; fuzzy attacks inject random IDs and data; impersonation attacks impersonate legitimate messages (e.g., CAN ID '0×164'). The dataset contains real-time CAN traffic and is, therefore, appropriate for assessing IDS models in real in-vehicle scenarios.

2.2. Data preprocessing

We used CAR hacking dataset and OTIDS dataset, which include fuzzy, DoS, RPM spoofing, and gear spoofing, impersonation attacks collected from in-vehicle network logs. Each dataset contains columns such as timestamp, data length code (DLC), CAN ID, data bytes, and an attack label. To evaluate the proposed method, a train-validation split was used during training, with a 20% data used for validation. To reduce computational load while maintaining representativity, a 10% random sample was extracted from each dataset respectively. Each dataset was then applied to have the columns standardized and had labels assigned to each based on the kind of attack it was (e.g. 'DoS attack' and 'fuzzy attack'). Since CAN IDs and data bytes were originally in hexadecimal, we converted it to integers to make it compatible with DL models.

A new 'message' feature was also created for feature engineering purposes by concatenating the eight data bytes (DATA [0] to DATA [7]) to form a single numerical value, which was subsequently saved as a separate feature. The timestamp column was also converted to a datetime format in case we needed to work in real time. Finally, standard scaling was applied to all features, normalizing data values so that each feature contributed equally to the learning process. The SMOTE was used to fix the dataset's class imbalance. Figure 5 shows the class distributions in the HCRL dataset before (Figure 5(a)) and after (Figure 5(b)) applying SMOTE and Figure 6 shows the OTIDS dataset before (Figure 6(a)) and after (Figure 6(b)) applying SMOTE. Each dataset was highly imbalanced, the imbalance can bias the model toward majority classes, leading to poor detection of rare intrusions. To address this, we applied SMOTE to the training data, generating additional synthetic samples for minority classes.

Figures 5 and 6 show the class distributions before and after SMOTE was used. SMOTE works well to balance class representation and make it easier to find attacks on minorities, but it may add synthetic samples that don't fully show the time and structure limits of real CAN traffic, especially for attacks that happen quickly or at low frequencies. Synthetic interpolation like this can smooth out sudden changes in attacks or create feature combinations that don't really represent real-world message sequences. We looked at other ways to deal with imbalance, like adaptive synthetic sampling (ADASYN), random undersampling, and cost-sensitive learning. ADASYN dynamically focuses on samples that are harder to learn, but it can also make noise in high-dimensional sequence data worse. Undersampling, on the other hand, runs the risk of throwing away useful benign patterns that are important for controlling false positives in safety-critical automotive environments. In this study, SMOTE was chosen as a balanced compromise because it is easy to use, stable, and widely used in the literature on CAN intrusion detection. It allows for controlled class balancing without losing too much information about the majority class. However, acknowledging the constraints of synthetic oversampling, subsequent research will explore sequence-aware augmentation methods, generative models, and hybrid sampling approaches that more effectively maintain temporal coherence and authentic CAN dynamics.

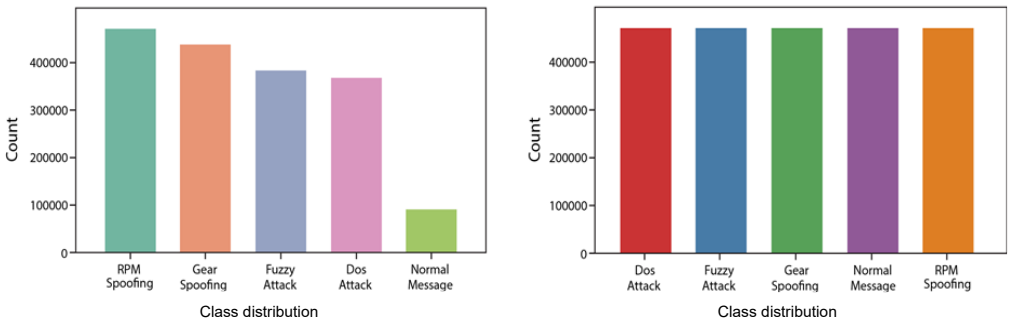


Figure 5. HCRL dataset of (a) before and (b) after SMOTE

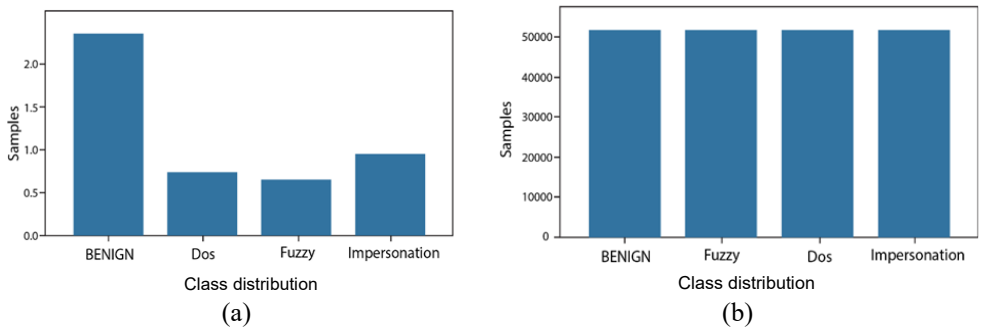


Figure 6. OTIDS dataset of (a) before and (b) after SMOTE

2.3. Model architecture

CNNs, GRUs, LSTM, and attention mechanisms are all included in the suggested hybrid DL model to efficiently identify temporal and spatial patterns in the data. A one-dimensional feature sequence is sent to

the input layer, where it is rearranged as necessary for CNN processing. A Conv1D layer with 128 filters and a kernel size of 3 is utilized to extract spatial features. The convolutional operation in a 1D CNN layer can be formally expressed as (1).

$$O[i] = \sum_{j=0}^{K-1} W[j] \cdot X[i+j] + b \quad (1)$$

Where $O[i]$ represents output at position i, W denotes the filter, K is the kernel size, X is the input sequence and b are the bias term. To generalize this operation for multi-channel inputs and multiple filters, the convolutional computation becomes (2).

$$y_i^{(k)} = f\left(\sum_{c=1}^{c} \sum_{j=1}^{k} w_j^{(k,c)} \cdot x_{i+j-1}^{(c)} + b^{(k)}\right) \quad (2)$$

Where $y_i^{(k)}$ is the output at position i for the k^{th} filter, $x^{(c)}$ is the input of c^{th} channel and $w_j^{(k,c)}$ is the filter weight for kernel position j, input channel c and output filter k. $b^{(k)}$ is the bias term for the k^{th} filter and $f(\cdot)$ is the activation function. Following this, a MaxPooling1D layer is applied to reduce spatial dimensions and computational load, computed as (3).

$$O[i] = max(X[i \cdot S : i \cdot S + P]) \quad (3)$$

Where S is the stride and P is the pool size, batch normalisation is employed to enhance convergence. In the proposed hybrid model, the temporal dependencies are captured through the LSTM and GRU layers using the following key equations. For the LSTM layer, the cell state c_t and the hidden state h_t as shown in (4) and (5).

$$c_t = f_t \cdot c_{t-1} + i_t \cdot tanh\left(W_c \cdot \left[h_{t-1}, x_t\right] + b_c\right) \quad (4)$$

$$h_t = o_t \cdot tanh(c_t) \quad (5)$$

Where f_t,i_t and o_t represent the forget, input, and output gates, respectively. These gates control the flow of information, enabling the model to retain relevant information over time and discard irrelevant data. The GRU layer refines these temporal features further, with the hidden state h_t modified as (6).

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \widetilde{h_t} \quad (6)$$

Where $\tilde{h}_t = tanh\,(W_h \cdot \left[r_t \cdot h_{t-1}, x_t\right] + b_h)$ is the candidate hidden state, and z_t is the update gate. This formulation enables the GRU layer to efficiently capture dependencies without the explicit memory cell used in LSTMs, making it computationally efficient while retaining essential temporal information. The attention scores and context vector can be computed as (7) and (8).

Attention weights: $a_t = softmax(W_a \cdot h_t)$ (7)

Where W_a is the weight matrix for attention.

Context vector: $c_t = \sum_{i=1}^{T} a_{ti} \cdot h_i$ (8)

Where a_{ti} are the attention weights for each hidden state h_i. T is the total number of time steps. The softmax function is used to convert the logits to probabilities as shown in (9).

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (9)$$

Where $P(y_i)$ is the predicted probability for class i. z_i are the logits from the final layer before applying softmax. K is the total number of classes. For multi-class classification, the categorical cross-entropy loss is calculated as (10).

$$L(y, \hat{y}) = -\sum_{i=1}^{k} y_i \, log(\hat{y}_i) \quad (10)$$

Where y is the true label (one-hot encoded). $\hat{y}$ is the predicted probability distribution from the softmax layer. The L2 regularization term to be added to the loss function is defined in (11).

$$L_{reg} = \lambda \sum_{j=1}^{N} W_j^2 \quad (11)$$

Where λ is the regularization parameter. W_j are the weights of the model.

We first extract spatial features using 3-by-128 filters in a Conv1D layer, followed by MaxPooling1D to compress spatial features and batch normalization to improve convergence. Before feeding the data into the model, the standardized CAN feature vectors are grouped into sliding windows of 10 consecutive messages (stride 1), and each window is treated as one input sequence in order to preserve temporal structure. A 64-unit LSTM layer with return sequences is introduced to capture temporal dependencies, with a dropout rate of 0.2 to prevent overfitting (as used in the final model). This is then followed by a 64-unit GRU layer with return sequences and a dropout rate of 0.2 to further process the output of the LSTM layer and obtain more accurate temporal features. The LSTM layer is used to capture long-range temporal dependencies in CAN traffic, but LSTMs are computationally heavy. So, placing a GRU layer after the LSTM allows more efficient refinement of these temporal features. The GRUs retain essential sequence patterns while reducing complexity due to their simpler gating mechanism. This improves convergence, limits overfitting, and makes the model more practical for real-time, ECU-constrained in-vehicle deployment. This is followed by an attention layer over the GRU output so that the network is able to focus on important parts of the sequence while making each prediction. The attention output is then passed through two fully connected layers with 128 and 64 units, respectively, each using L2 regularization (0.001) and dropout (0.3) to prevent overfitting. Finally, a softmax layer generates the class probabilities for each attack type to provide effective intrusion detection. The model was trained using the Adam optimizer (learning rate =0.001) with categorical cross-entropy loss for 50 epochs and a batch size of 128. Early stopping and learning-rate reduction callbacks were used to enhance convergence and avoid overfitting. The entire model was coded in Python 3.7 using TensorFlow and Keras. Training and testing were conducted on the shared NVIDIA Tesla K80 GPU in Google Colab; while preprocessing and initial development were performed on a local machine with an Intel Core-i5 processor and 8 GB RAM.

3. RESULTS AND DISCUSSION

This section considers the performance of the proposed hybrid DL approach. Both the HCRL and OTIDS datasets, containing varied attack types such as DoS, fuzzy, and impersonation, were utilized. We employed common performance metrics, including accuracy, precision, recall, and F1-score, and statistical validations (Tables 2 and 3) to evaluate the performance of the introduced hybrid DL model. Detailed performance results and metrics are presented in Table 4. The model showed enhanced detection capability with zero or negligible false positives and false negatives, with 96.30% accuracy on the HCRL CAR hacking dataset using weighted precision, recall, and F1-score of 0.96. Likewise, the model showed superb generalizability across different attack types, such as DoS, fuzzy, and impersonation attacks, and reached 99.78% accuracy, 0.99 precision, 0.99 recall, and 0.99 F1-score on the OTIDS dataset. The confusion matrices, illustrating the classification performance for each class, are presented in Figure 7, where Figure 7(a) shows the HCRL and Figure 7(b) shows the OTIDS. A comparative analysis of precision, recall, and F1-score for both datasets are shown in Figure 8, where Figure 8(a) shows the OTIDS and Figure 8(b) shows the HCRL. Additionally, the ROC curves for both datasets, shown in Figure 9, where Figure 9(a) shows the OTIDS and Figure 9(b) shows the HCRL, demonstrate the high discriminative power of the model.

Table 2. OTIDS dataset statistical significance results

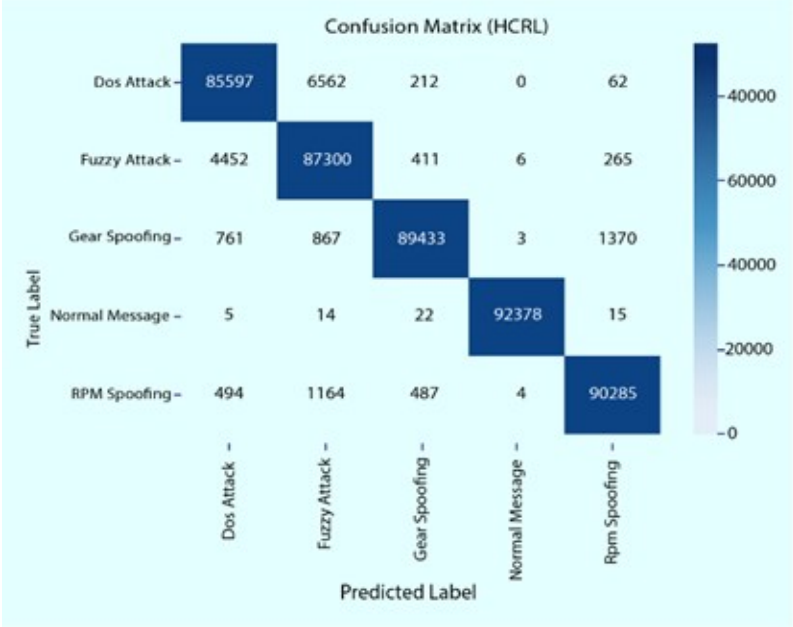
Comparison	DriveShield accuracy	Baseline accuracy	Improvement	Paired-t p-value	Wilcoxon p-value	Significance
vs CNN	0.9979	0.9420	+0.0559	1.07×10^{-83}	7.55×10^{-10}	Significant
vs LSTM	0.9979	0.8420	+0.1559	3.51×10^{-95}	7.55×10^{-10}	Significant
vs GRU	0.9979	0.8190	+0.1789	1.55×10^{-98}	7.55×10^{-10}	Significant

Table 3. HCRL dataset statistical significance results

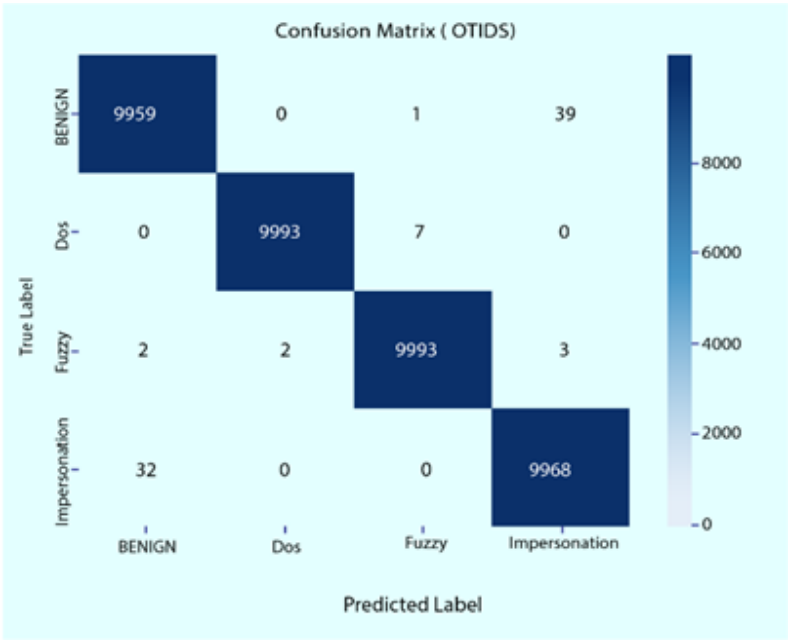
Comparison	DriveShield accuracy	Baseline accuracy	Improvement	Paired-t p-value	Wilcoxon p-value	Significance
vs LSTM	0.9630	0.9010	+0.0620	4.15×10^{-112}	7.55×10^{-10}	Significant
vs GRU	0.9630	0.9103	+0.0527	2.19×10^{-106}	1.78×10^{-15}	Significant
vs CNN	0.9630	0.8993	+0.0637	2.35×10^{-111}	7.55×10^{-10}	Significant

Table 4. Results of DriveShield

Dataset	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
HCRL car hacking	96.30	96	96	96
OTIDS	99.78	99.78	99.78	99.78



(a)



(b)

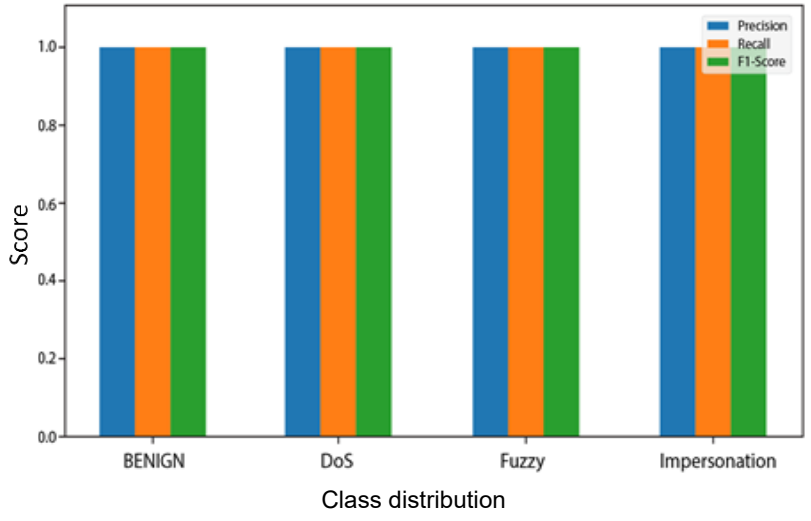
Figure 7. Confusion matrix of (a) HCRL and (b) OTIDS

We used per-class metrics, confusion matrices, ROC curves, and statistical significance tests on both the HCRL and OTIDS datasets to see how well the proposed DriveShield model worked. The class-level evaluation shows that DriveShield always gets high precision, recall, and F1-scores for all types of attacks. It is especially good at finding gear spoofing and RPM spoofing attacks in the HCRL dataset and almost perfect

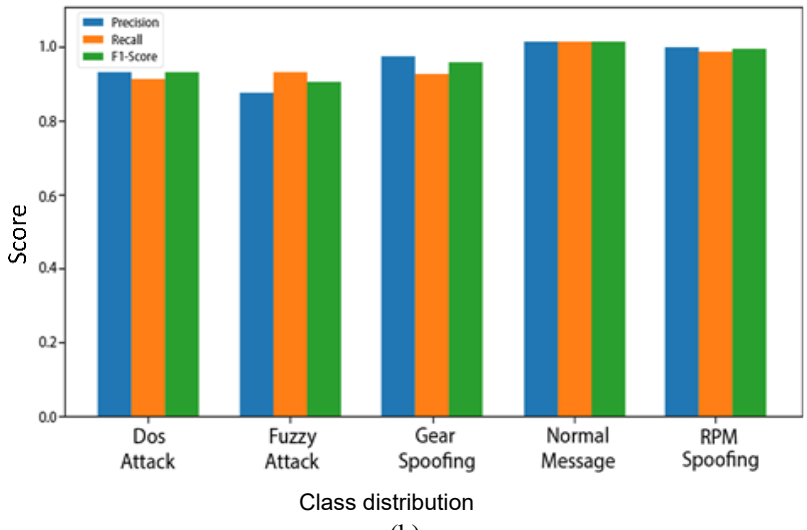
at finding DoS and fuzzy attacks in the OTIDS dataset. This shows that the model can tell the difference between real CAN messages and malicious injections with very few false positives.

We used non-parametric bootstrap resampling on the test set predictions to find 95% confidence intervals. The model's accuracy on the HCRL dataset was 96.98% (95% CI: 96.94%–97.03%), and its macro-F1-score was 0.9699 (95% CI: 0.9695–0.9704). The model also did very well on the OTIDS dataset, with an accuracy of 99.72% (95% CI: 99.69%–99.74%) and a macro-F1-score of 0.9972 (95% CI: 0.9969–0.9974%). The small confidence intervals in both datasets show that the performance gains are not just a fluke. These results are great, but there are still some problems with real-world in-vehicle intrusion detection that make it hard to find low-frequency and stealthy attacks.

In the original OTIDS dataset, a lot of the raw CAN traces are from benign traffic. Impersonation and fuzzy attacks, on the other hand, happen less often and don't last as long. To try to fix this imbalance during training, class balancing and window-level sequence construction were used, but these methods might not fully show the differences and lack of patterns in rare attacks that happen in real-life driving situations. The HCRL dataset also has a lot of message injection attacks, but small time overlaps between benign and attack sequences can make it harder to remember quick, short-lived intrusions. Minor variations between the point estimates reported in Table 4 and the bootstrap-based accuracy values arise from resampling-based estimation and metric aggregation differences, which is a common and acceptable behavior in statistical performance analysis.

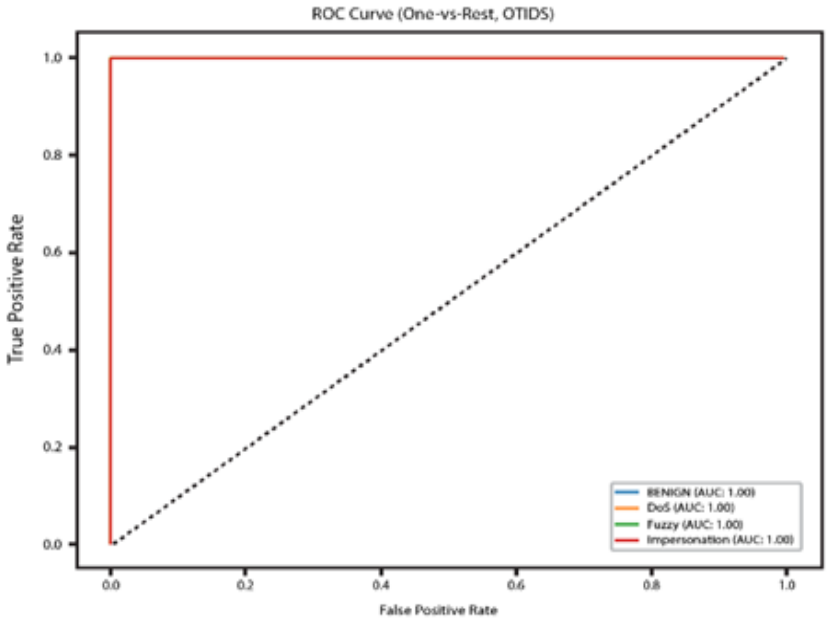


(a)

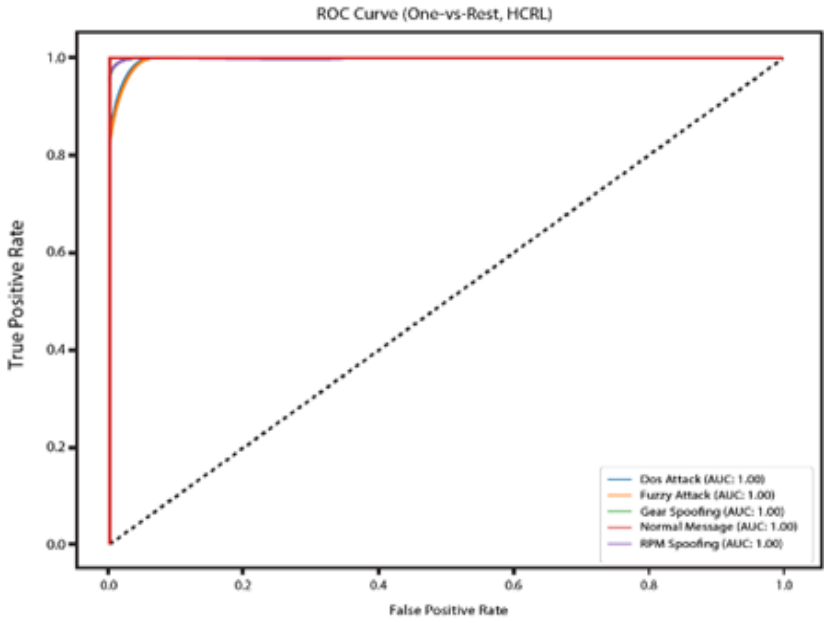


(b)

Figure 8. Precision, recall, F1 score of (a) OTIDS and (b) HCRL



(a)



(b)

Figure 9. ROC curve of (a) OTIDS and (b) HCRL dataset

3.1. Statistical validation and paired significance analysis

The DriveShield achieved the highest intrusion detection performance across both datasets. These results confirm that the accuracy gains are not due to random variation and the hybrid architecture provides consistent superiority in learning CAN bus attack patterns. This strongly validates DriveShield's ability to model both local feature correlations CNN, and long-range temporal dependencies (LSTM + GRU) in vehicular communication. The paired t-test and the Wilcoxon signed-rank test both show that the proposed hybrid architecture is statistically better than the CNN-, LSTM-, and GRU-only baselines in both datasets ($p < 0.01$).

3.2. Hyperparameter selection and tuning

To validate the architectural and optimization decisions of the proposed hybrid CNN, LSTM, GRU, and attention model, a methodical hyperparameter tuning process was executed utilizing a controlled grid

search over critical parameters influencing spatial–temporal feature learning and computational efficiency. The search space included the number of convolutional filters (64, 128), the sizes of the kernels (3, 5), the sizes of the recurrent units (64, 128), the rates of recurrent dropout (0.2, 0.3), and the learning rates of the Adam optimizer (0.001, 0.0005). We used the validation macro-F1 score to choose the best model. This score gives a fair evaluation across all attack classes and works well with intrusion detection datasets that aren't balanced. Kernel size 3 consistently outperformed larger kernels in both the HCRL and OTIDS datasets.

This shows that short-range temporal patterns in CAN traffic are better at detecting intrusions. Increasing the number of convolutional filters to 128 made the features more expressive without making the training less stable. On the other hand, using larger kernel sizes and higher dropout values made the performance only slightly worse. Configurations with higher recurrent capacity (e.g., 128 units) got similar validation macro-F1 scores, but they also made the computation more expensive and the convergence less stable. So, the final model configuration was chosen based on the best balance between detection performance, convergence stability, and computational efficiency needed for real-time ECU deployment. A recurrent unit size of 64, a dropout rate of 0.2, and a learning rate of 0.001 worked well for generalizing across datasets while keeping the model simple. Table 5 summarizes the top-ranked hyperparameter configurations obtained during the grid search for both datasets.

Table 5. Hyperparameter configurations evaluated using validation macro-F1-score

Dataset	Rank	Conv filters	Kernel size	RNN units	Dropout	Learning rate	Val macro-F1
HCRL	1	128	3	128	0.2	0.0010	0.96279
HCRL	2	64	3	64	0.2	0.0010	0.96223
HCRL	3	128	3	64	0.2	0.0005	0.96179
HCRL	4	128	3	64	0.2	0.0010	0.96173
HCRL	5	128	3	64	0.3	0.0010	0.96116
HCRL	6	128	5	64	0.2	0.0010	0.95985
OTIDS	1	128	3	128	0.2	0.0010	0.99716
OTIDS	2	128	3	64	0.3	0.0010	0.99642
OTIDS	3	64	3	64	0.2	0.0010	0.99520
OTIDS	4	128	3	64	0.2	0.0010	0.99462
OTIDS	5	128	3	64	0.2	0.0005	0.99365
OTIDS	6	128	5	64	0.2	0.0010	0.99217

To validate these results, we compared our results with some recent literature works. For example, Ahmed *et al.* [18] applied a CNN (VGG-16)-based image approach on CAN datasets and attained around 95–96% accuracy, but without temporal modeling and real-time adaptability. Alferaidi *et al.* [19] applied a CNN–LSTM model on the NSL-KDD dataset with ~99.7% accuracy but without handling class imbalance and real CAN traffic understanding. Kang and Kang [26] proposed a DNN for CAN traffic but without hybrid architectural integration and class-level performance measures. Our solution, in contrast to these studies, employs a combination of sequence learning layers (LSTM and GRU) and attention mechanisms, incorporates SMOTE class balancing, and is evaluated on two public CAN datasets with thorough metrics and visualizations (e.g., confusion matrices and ROC curves). Class level results, tabulated in Table 6, indicate that our approach provides an optimal balance of detection accuracy, minimum false positive rate, and ability to generalize with computational efficiency to utilize GPU-based models. Not only do these results establish the strength of the proposed IDS, but they also reflect its originality in learning more precise spatial-temporal features than most of the current single-architecture approaches.

Although DriveShield shows a very high overall detection capability, the detection of low-frequency and stealthy CAN-bus attacks is inherently challenging. Such attacks, like DoS and fuzzy injections, appear very seldom when compared to normal traffic patterns in real automotive environments [29]. Their detection will be improved with more volumes of real attack traffic, continuous learning pipelines, and, importantly, adversarial or generative data augmentation that could provide more realistic and diverse malicious patterns than traditional oversampling. By incorporating incremental learning and richer threat visibility, future iterations of DriveShield can further enhance resilience against sophisticated, low-frequency intrusions in modern automotive networks. Transformer-based IDS models have recently emerged for CAN intrusion detection, leveraging self-attention to model long-range temporal dependencies. While architectures such as CAN-BERT and temporal transformer IDS [30] variants demonstrate strong detection capability, they typically require significantly higher computational resources, GPU acceleration, and longer inference time compared to recurrent hybrid models. This makes direct deployment on automotive ECUs difficult without aggressive model compression.

To the best of our knowledge, existing CAN-bus IDS studies predominantly focus on offline classification accuracy, with limited consideration of deployability on automotive ECUs. Beyond proposing a

novel hybrid CNN–LSTM–GRU–attention architecture, this work evaluates real-time feasibility by benchmarking end-to-end inference latency and outlining a practical embedded deployment roadmap. In an unoptimized FP32 TensorFlow implementation, the proposed model achieves a mean inference latency of approximately 105 ms per input sequence (window size =10 CAN messages), with P95 latency below 160 ms, measured on a shared NVIDIA Tesla K80 GPU environment. Similar latency ranges have been reported in recent on-device CAN intrusion detection studies, where optimized and quantized DL models achieved near real-time performance on resource-constrained platforms such as Raspberry Pi [31]. The deployment plan targets <10 MB memory and <10 ms inference through structured optimizations, including structured pruning, post-training INT8 quantization with calibration, TensorRT or mixed-precision acceleration, and sequence-length refinement. Crucially, these optimizations are not applied to the reported FP32 evaluation results, ensuring that accuracy remains fully preserved. By combining high-accuracy detection with deployment-oriented analysis, this work bridges the gap between academic IDS research and real in-vehicle security implementations.

Table 6. Class-level performance metrics for HCRL and OTIDS

Dataset	Attack type	Precision (%)	Recall (%)	F1-score (%)
HCRL	DoS attack	94	93	93
	Fuzzy attack	91	94	93
	Gear spoofing	99	97	98
	RPM spoofing	98	98	98
	BENIGN	100	100	100
OTIDS	BENIGN	99.66	99.66	99.66
	DoS	99.98	99.98	99.98
	Fuzzy	99.92	99.92	99.92
	Impersonation	99.58	99.58	99.58

4. CONCLUSION

We presented DriveShield, an adaptive IDS that uses a hybrid DL architecture to protect communications on the CAN in vehicles. The proposed model can find both spatial patterns and temporal dependencies in CAN traffic by combining convolutional layers with LSTM and GRU networks and an attention mechanism. Testing on the HCRL and OTIDS datasets shows that DriveShield consistently and reliably detects multiple types of attacks, with high accuracy and a balanced precision–recall behavior. These results show how useful it is to combine temporal modeling and attention-based feature weighting for detecting intrusions in vehicles. At the same time, the study reveals challenges in identifying low-frequency and short-duration attacks under highly imbalanced traffic conditions, and the current evaluation has been limited to controlled experimental settings. Future work will concentrate on enhancing the framework to incorporate unsupervised and self-supervised learning, facilitating the management of evolving and novel attacks. Additionally, it will explore lightweight model variants and efficient attention mechanisms appropriate for implementation on resource-limited automotive ECU.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Vismaya Kootayi Kunnacheri	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
Arul Leena Rose Peter Joseph					✓	✓				✓		✓	✓	

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The author declares that there are no known conflicts of interest associated with this publication. There are no financial or personal relationships that could inappropriately influence or bias the content of this work.

DATA AVAILABILITY

All datasets used in this study (HCRL and OTIDS) are publicly available through the sources cited in the References. The code used in this study are not in public domain because of institutional limits. They can be shared by the corresponding author if the request is fair and reasonable, and always in line with the findable, accessible, interoperable, and reusable (FAIR) data principles.

REFERENCES

[1] U. Ahmed, J. C.-W. Lin, and G. Srivastava, "Privacy-preserving deep reinforcement learning in vehicle ad hoc networks," *IEEE Consumer Electronics Magazine*, vol. 11, no. 6, pp. 41–48, Nov. 2022, doi: 10.1109/MCE.2021.3088408.

[2] V. S. Barletta, D. Caivano, A. Nannavecchia, and M. Scalera, "Intrusion detection for in-vehicle communication networks: an unsupervised kohonen SOM approach," *Future Internet*, vol. 12, no. 7, 2020, doi: 10.3390/FI12070119.

[3] K. Gu, X. Ouyang, and Y. Wang, "Malicious vehicle detection scheme based on spatio-temporal features of traffic flow under cloud–fog computing-based IoVs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 9, pp. 11534–11551, 2024, doi: 10.1109/TITS.2024.3369974.

[4] N. Alkhatib, M. Mushtaq, H. Ghauch, and J. L. Danger, "Unsupervised network intrusion detection system for AVTP in automotive Ethernet networks," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2022, pp. 1731–1738, doi: 10.1109/IV51971.2022.9827285.

[5] L. Zhang and D. Ma, "A hybrid approach toward efficient and accurate intrusion detection for in-vehicle networks," *IEEE Access*, vol. 10, pp. 10852–10866, 2022, doi: 10.1109/ACCESS.2022.3145007.

[6] Y. Zeng, M. Qiu, D. Zhu, Z. Xue, J. Xiong, and M. Liu, "DeepVCM: a deep learning-based intrusion detection method in VANET," in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, May 2019, pp. 288–293, doi: 10.1109/BigDataSecurity-HPSC-IDS.2019.00060.

[7] O. Y. Al-Jarrah, C. Maple, M. Dianati, D. Oxtoby, and A. Mouzakitis, "Intrusion detection systems for intra-vehicle networks: a review," *IEEE Access*, vol. 7, pp. 21266–21289, 2019, doi: 10.1109/ACCESS.2019.2894183.

[8] Y. Gao, H. Wu, B. Song, Y. Jin, X. Luo, and X. Zeng, "A distributed network intrusion detection system for distributed denial of service attacks in vehicular ad hoc network," *IEEE Access*, vol. 7, pp. 154560–154571, 2019, doi: 10.1109/ACCESS.2019.2948382.

[9] A. Alsaleh, "A novel intrusion detection model of unknown attacks using convolutional neural networks," *Computer Systems Science and Engineering*, vol. 48, no. 2, pp. 431–449, 2024, doi: 10.32604/csse.2023.043107.

[10] R. Dasgupta, M. Pramanik, P. Mitra, and D. R. Chowdhury, "Intrusion detection for power grid: a review," *International Journal of Information Security*, vol. 23, no. 2, pp. 1317–1329, 2024, doi: 10.1007/s10207-023-00789-6.

[11] B. S. Bari, K. Yelamarthi, and S. Ghafoor, "Intrusion detection in vehicle controller area network (CAN) bus using machine learning: a comparative performance study," *Sensors*, vol. 23, no. 7, 2023, doi: 10.3390/s23073610.

[12] R. Rai, J. Grover, P. Sharma, and A. Pareek, "Securing the CAN bus using deep learning for intrusion detection in vehicles," *Scientific Reports*, vol. 15, no. 1, 2025, doi: 10.1038/s41598-025-98433-x.

[13] K. Bansal and A. Singhrova, "Review on intrusion detection system for IoT/IIoT -brief study," *Multimedia Tools and Applications*, vol. 83, no. 8, pp. 23083–23108, 2024, doi: 10.1007/s11042-023-16395-6.

[14] H. Yan, X. Lin, S. Li, H. Peng, and B. Zhang, "Global or local adaptation? client-sampled federated meta-learning for personalized IoT intrusion detection," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 279–293, 2025, doi: 10.1109/TIFS.2024.3516548.

[15] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, 2020, doi: 10.1016/j.vehcom.2019.100198.

[16] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "LSTM-based intrusion detection system for in-vehicle can bus communications," *IEEE Access*, vol. 8, pp. 185489–185502, 2020, doi: 10.1109/ACCESS.2020.3029307.

[17] H. Ma, J. Cao, B. Mi, D. Huang, Y. Liu, and S. Li, "A GRU-based lightweight system for CAN intrusion detection in real time," *Security and Communication Networks*, 2022, doi: 10.1155/2022/5827056.

[18] I. Ahmed, G. Jeon, and A. Ahmad, "Deep learning-based intrusion detection system for internet of vehicles," *IEEE Consumer Electronics Magazine*, vol. 12, no. 1, pp. 117–123, 2023, doi: 10.1109/MCE.2021.3139170.

[19] A. Alferaidi *et al.*, "Distributed deep CNN–LSTM model for intrusion detection in IoT-based vehicles," *Mathematical Problems in Engineering*, 2022, doi: 10.1155/2022/3424819.

[20] N. Alkhatib, H. Ghauch, and J. L. Danger, "SOME/IP intrusion detection using deep learning-based sequential models in automotive Ethernet networks," in *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2021, pp. 954–962, doi: 10.1109/IEMCON53756.2021.9623129.

[21] H. Alqahtani and G. Kumar, "A deep learning-based intrusion detection system for in-vehicle networks," *Computers and Electrical Engineering*, vol. 104, 2022, doi: 10.1016/j.compeleceng.2022.108447.

[22] A. Anzer and M. Elhadef, "A multilayer perceptron-based distributed intrusion detection system for internet of vehicles," in *Proceedings - 4th IEEE International Conference on Collaboration and Internet Computing (CIC)*, 2018, pp. 438–445, doi: 10.1109/CIC.2018.00066.

[23] J. Ashraf, A. D. Bakhshi, N. Moustafa, H. Khurshid, A. Javed, and A. Beheshti, "Novel deep learning-enabled LSTM autoencoder architecture for discovering anomalous events from intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4507–4518, 2021, doi: 10.1109/TITS.2020.3017882.

[24] X. J. Li, M. Ma, and Y. Sun, "An adaptive deep learning neural network model to enhance machine-learning-based classifiers for intrusion detection in smart grids," *Algorithms*, vol. 16, no. 6, 2023, doi: 10.3390/a16060288.

[25] M. M. El-Gayar, F. A. F. Alrslani, and S. El-Sappagh, "Smart collaborative intrusion detection system for securing vehicular networks using ensemble machine learning," *Information*, vol. 15, no. 10, 2024, doi: 10.3390/info15100583.
[26] M. J. Kang and J. W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," *PLoS ONE*, vol. 11, no. 6, 2016, doi: 10.1371/journal.pone.0155781.
[27] Hacking and Countermeasure Research Lab (HCRL), "Car-hacking dataset," *ocslab.hksecurity.net*, 2020. Accessed: Sep. 24, 2024. [Online]. Available: https://ocslab.hksecurity.net/Datasets/car-hacking-dataset
[28] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDS: a novel intrusion detection system for in-vehicle network by using remote frame," in *Proceedings - 2017 15th Annual Conference on Privacy, Security and Trust (PST)*, 2018, pp. 57–66, doi: 10.1109/PST.2017.00017.
[29] B. Lampe and W. Meng, "Can-train-and-test: a curated CAN dataset for automotive intrusion detection," *Computers and Security*, vol. 140, 2024, doi: 10.1016/j.cose.2024.103777.
[30] T. P. Nguyen, H. Nam, and D. Kim, "Transformer-based attention network for in-vehicle intrusion detection," *IEEE Access*, vol. 11, pp. 55389–55403, 2023, doi: 10.1109/ACCESS.2023.3282110.
[31] S. Rajapaksha, H. Kalutarage, M. O. Al-Kadri, A. Petrovski, and G. Madzudzo, "Improving in-vehicle networks intrusion detection using on-device transfer learning," in *Symposium on Vehicles Security and Privacy (VehicleSec), San Diego, USA*, 2023, doi: 10.14722/vehiclesec.2023.23088.

BIOGRAPHIES OF AUTHORS

Vismaya Kootayi Kunnacheri holds a Master of Computer Applications degree from APJ Abdul Kalam Technological University, Kerala, in 2021. She has qualified the University Grants Commission National Eligibility Test (UGC NET) for assistant professor. She also received her B.Sc. (Computer Science) from Kannur University, Kerala, in 2019. She is currently a Ph.D. scholar at the Computer Science Department in SRM Institute of science and technology, Chennai, Tamil Nadu. Her research includes deep learning, vehicle security, machine learning, cyber security. She can be contacted at email: vismayakootayi@gmail.com.

Arul Leena Rose Peter Joseph holds a Ph.D. degree in Computer Science from Mother Theresa University, Kodaikanal, India in 2016. She is currently working as a professor and HOD in the Department of Computer Applications, SRM Institute of Science and Technology. She has 25 years of teaching experience and 7 years of research experience and is also proficient in many technical areas, such as software engineering, DBMS, and artificial intelligence. Her research areas include machine learning, image processing, and artificial neural networks. She has published various research papers in International Journals and conferences, which are indexed in Scopus and Web of Science. She can be contacted at email: leena.rose527@gmail.com.