(R3: 1) Unifying Safety and Security: Real-Time Detection of Driving Aggressions and Cyber-Attacks via Vehicular CAN Analysis

Md Rezanur Islam, Mahdi Sahlabadi (Senior Member, IEEE), Munkhdelgerekh Batzorig, Kangbin Yim

***Abstract*—Aggressive driving is a leading cause of accidents. Meanwhile, modern vehicles also face an increasing attack surface due to enhanced connectivity. Since driving behavior is often habitual, an effective detection system can improve safety by identifying risky actions and cyber-attacks, and the development of such systems has the potential to transform driving habits for long-term improvements. The current study presents a unified system for the real-time detection of abnormal driving behaviors and cyber-attacks within in-vehicle networks (IVNs). Our system uses minimal CAN-specific functional data to identify aggressive actions such as abrupt braking, lane changes, acceleration, and cyber-attacks. This system uniquely distinguishes between driver-induced aggression and attack- triggered behavior, thus ensuring the precise detection of human errors and external threats while operating entirely on CAN data without the use of any external sensors (R3: 1) (Safety-Critical Cybersecurity). Our approach leverages a diverse dataset encompassing various driving scenarios, road conditions, and driver profiles to ensure reliability. The proposed system employs a specially designed deep learning architecture based on L-CNN (Lightweight CNN) and (R2, R3: 1, 5) implemented on Raspberry Pi 4, which is optimized for resource-constrained automotive environments. This approach achieves near real-time detection time with minimal feature extraction, leading to a substantial reduction in computational overhead while ensuring high precision and low latency. Time-series analysis of CAN data enables the accurate classification of driving behaviors and the detection of cyber intrusions, reaching an overall accuracy of 99%, along with a threefold improvement in detection time compared to earlier studies. This holistic solution enhances vehicle robustness against evolving cyber-physical threats, ultimately offering a cost-effective, energy-efficient method to ensure safety and security in modern vehicles.**

***Index Terms*—Aggressive Driving Detection, Safety, Cybersecurity, IVN, CAN.**

I. Introduction

(R3: 1)

SAFETY is the first and most important requirement in driving, and aggressive driving is a major threat to road safety. At the same time, cybersecurity threats in modern vehicles can also compromise driving safety; therefore, this study refers to such risks as Safety-Critical Cybersecurity. Aggressive driving, which is characterized by actions such as abrupt braking, lane changes, and acceleration, is dangerous and significantly threatens road safety [1], [2], [3], [4]. AI-enhanced automated solutions use mobile devices, video, GPS, and external sensors to detect aggressive driving behaviors [5], [6]. However, few studies have looked into using vehicle internal sensors, like Controller Area Network (CAN) data, for this purpose [7], [8], [9], with a particularly suitable method involving the use of ' low-layer' CAN frames [10]. Aggressive driving behaviors can be replicated by cyber attackers sending commands through the CAN bus within a vehicle's network. These Cyber-Physical Driving [11] attacks are difficult to detect when using only external data, but CAN data is widely available across all types of vehicles, and it can be easily accessed. By contrast, data from other sensors often comes with high costs and installation complexity [12], and they are mostly unable to detect behavioral features in real-time[1].

The CAN bus provides real-time data on speed, braking, and engine performance, which allows CAN to effectively monitor driving behavior and enhance vehicle security against cyber threats [8], [9]. Various vehicle models are equipped with different and several numbers of Electronic Control Units (ECUs) [14] that communicate as shown in Fig. 1 and generate significant amounts of binary values represented in hexadecimal format within a short time interval. (R3: 2) For example, Kia Soul (2017) and Tesla S3 (2020) models has been shown to generate roughly 260,000 and 400,000 bits per second [15]. Such massive data generation leads to significant network traffic, as each ECU receives all data since the CAN bus broadcasts every single message [16]. A thorough understanding of CAN, and particularly of which IDs correspond to which function, is essential for interpreting this large volume of raw data. Manufacturers often use a CAN database container, known as a DBC file, to store the CAN message configurations for a specific vehicle. However, these DBC files are not publicly available to researchers [17]. To overcome the challenges involved in interpreting large volumes of raw CAN data and the lack of access to DBC files, we used fuzzing tests [18] and frequency analysis to identify function IDs and their payloads.

The increasing complexity of IVNs in modern vehicles poses challenges for maintaining safety and security [19], which are often handled separately, thus leading to increased

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Convergence security core talent training business support program (IITP-2024-2710008611) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation) and Soonchunhyang University Research Fund. (Corresponding author: Kangbin Yim.)

Md Rezanur Islam, Mahdi Sahlabadi, Munkhdelgerekh Batzorig and Kangbin Yim are with the Department of Software Convergence and the Department of Information Security Engineering, Soonchunhyang University, Asan-si, Korea (arupreza@sch.ac.kr; sahlabadi@sch.ac.kr; munkh@sch.ac.kr; yim@sch.ac.kr).

[1]Near real-time driving behavior could be counted in one millisecond [13].

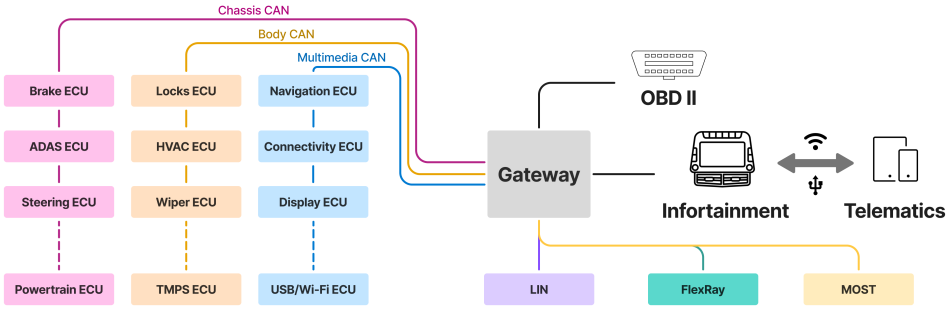


Fig. 1: Overview of In-Vehicle Network (IVN) Architecture with CAN Bus Integration and ECU Inter-connectivity.

computational demands, costs, and inefficiencies. The reliability of any system in this context also depends on the use of precise labeling and a diverse range of IVN data, which is affected by different road conditions and driving styles [20]. While there have been some studies using CAN data, these approaches have involved complex feature extraction, preprocessing, and algorithms, resulting in higher detection times, which could have critical consequences for safety and security systems. Existing solutions face multiple challenges, such as difficulties integrating safety measures with cybersecurity protocols, overloaded vehicle processing, the introduction of latency, and a lack of access to well-defined and diverse data, altogether resulting in both false positives in dynamic situation and delays. To address these issues, we propose an integrated system that unifies safety and security by leveraging correlated CAN data features through a unified set of input features and employing a customized L-CNN (Lightweight CNN) deep learning algorithm. This approach is applied to data collected from various real-world driving situations, including urban, highway, city, and restricted environments, as well as from a diverse group of drivers spanning different age groups and genders. Our approach is expected to minimize preprocessing, reduce computational load, and enable the fast, accurate detection of abnormal behaviors and cyberattacks, ultimately ensuring efficient performance without straining vehicle resources.

II. Related Works

In this section, we highlight the recurring limitations in prior research on aggressive-driving detection, lightweight deep-learning models.

A. Aggressive Driving Detection: Existing Approaches

Vehicle accidents are often caused by abrupt braking, lane changes, and acceleration, which our research identifies as key aggressive driving behaviors [25]. It is crucial to prioritize both detection time and accuracy in detecting these behaviors; delayed detection can prevent timely intervention before an accident [26] whereas accuracy ensures reliable decision-making. To ensure the safety of both the driver and passengers, the detection system must process and respond in real-time, typically within milliseconds. Table I presents several studies that have explored the use of different chunk sizes, input features, and data preprocessing methods, all of which are found to have a direct impact on detection time. Achieving minimal detection times requires the use of smaller input chunks and lightweight data preprocessing and algorithms. However, the use of smaller chunks may lead to a higher false positive rate, as limited data may lack sufficient information to enable accurate predictions [27], so chunk size should be well-balanced. Cyber attackers can also initiate aggressive driving by injecting malicious data into the vehicle's system, in a process using minimal amounts of data at low frequencies, which makes it difficult to differentiate from legitimate actions. To the best of our knowledge, there have been no published studies distinguishing between aggressive driving that is initiated by a human driver and that which is initiated by a cyber attacker.

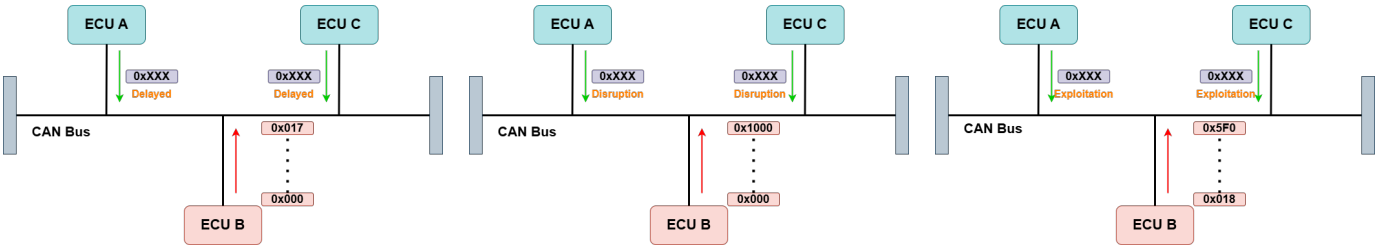
All studies have achieved high accuracy, as can be seen in Table I. However, most of the works in those studies have high detection times. High detection times when detecting aggressive driving could lead to delays in critical interventions, thus increasing the risk of accidents and endangering the safety of the vehicle's driver and passengers. To address this issue, some studies have utilized overlapping data chunks to ensure continuous monitoring and faster detection of aggressive driving behaviors, thereby achieving reduced detection times. However, the consideration of extensive and external features and complex feature extraction preprocessing steps—such as statistical operations, data conversion, OTA (Over The Air), and information fusion—can increase computational overhead along with detection time. On the other hand, it is essential to minimize computational load to achieve better real-time applicability [13].

In our proposed system as shown in Table I, we aim to minimize the number of features, perform minimal preprocessing, and optimize operational flow while increasing data diversity to reduce computational overhead and detection time as well as enhance system reliability. Most studies dealing with safety systems that include CAN data use OCSLab data [28]. However, this data often involves similar routes, which could lead to reliability issues when applied to dynamic routing scenarios. On the other hand, the dataset created explicitly for driver personalization tasks, as mentioned in [28], may not be suitable for use with a safety system. Shahverdy et al. achieved

[2]For unavailable data source, the chunk size and detection time are calculated based on the specifications of the Kia Soul.

TABLE I: Summary of driving behavior detection studies based on CAN, highlighting the algorithm, features, detection times, operational flow and accuracy across different datasets.

Study	Algorithm	Number of Features / Features	Feature Extraction Technique	Operational Flow	Chunk Size (Ms)	Overlap on Chunk	detection time (Ms)	Accuracy	Dataset / Num. of Drivers
Lattanzi et al. [9]	SVM, NN	Six / Speed, RPM, Brake, Engine Load, Throttle Position, Steering Wheel Angle	Statistical descriptors, hjorth parameters, kurtosis and skewness	Perform mathematical steps, then input the results into the classifier	20000 ms	50%	10000 ms	90%	Kia / OCSLab / 10
Fugiglando et al. [7]	PCA, K-means	Eight / Speed, RPM, Brake, Gas Pedal Position, Steering Wheel Angle, Steering Wheel Momentum, Time Gap, Acceleration	Raw data and statistical descriptors converted into histogram	Perform mathematical steps, convert data into a different form, then input into the classifier	60000 ms	0%	60000 ms	Not Specified	Audi / Own Data / 64
Kumar et al. [8]	Random Forest	Fifty / Speed, Brake, Fuel Consumption, Steering, Gear with several features	Raw data from correlation matrices	Raw data input into the classifier	None	0%	Near-real-time	100%	Kia / OCSLab / 10
Mohammed et al. [21]	Rule-based analysis with LabVIEW interface	Four / Speed, RPM, Engine Coolant Temperature and GPS	Category wise masking specific function payload	Convert data into a different form and data transmission through OTA into the system.	1000 ms^2	0%	1000 ms^2	Not Specified	Not Specified
Shahverdy et al. [22]	CNN	Five / Speed, RPM, Acceleration and Gravity (Smartphone), Throttle Position	Raw data and smartphone sensor data converted to image by recurrence plot	Convert data into a different form for information fusion input into the classifier.	50 ms	98%	Near-real-time	99.99%	Not Specified / Own Data / 3
Khosravinia et al. [23]	GConvLSTM	Thirty / Speed, Brake, Throttle position with several features	Pearson correlation forming a weighted adjacency matrix	Perform mathematical steps, convert data into a different form and data transmission through OTA into the classifier	10000 ms	50%	5000 ms	98%	Kia / OCSLab / 10
David et al. [12]	Fuzzy-macro LSTM	Six / Speed, RPM, Engine Coolant Temperature, Acceleration and time with fuzzy rules output through OTA	Raw data with 3D matrix and fuzzy rules	Raw data and Rule-based extracted features input through OTA into the classifier	500 ms^2	0%	500 ms^2	Not Specified	Not Specified / 1000
Zhang et al. [24]	Attention-based DeepConvGRU	Fifty one / Speed, Brake, Fuel Consumption, Steering, Gear with several features	Raw data	Raw data input into the classifier	30 ms	90%	Near-real-time	98%	Kia / OCSLab / 10
(R2:2) Rezanur et al. (Proposed Model)	L-CNN	Six / Six / Speed, RPM, Brake, Gear, Ultrasonic Sensor, High-priority Multi-function	Minimal preprocessing: raw CAN data, label encoding, min–max normalization	Direct raw-data input with 2D tensor (1×20×10)	20 ms	50%	20 ms^2	99%	Kia and Tesla / 16 and 6



(a) DoS (b) Fuzzing (c) Replay

Fig. 2: Attack Scenarios in In-Vehicle Networks by a Malicious ECU (ECU B). The figure shows four attack types: (a) DoS – ECU B blocks traffic by sending high-priority messages; (b) Fuzzing – ECU B injects malformed messages to disrupt ECUs; (c) Replay – ECU B reuses captured messages to trigger unauthorized actions.

high accuracy and detection time with a diverse dataset, but they did not consider aggressive driving behavior initiated by attackers. As presented in Table I, studies have shown that using model inputs that are standard features such as speed, RPM, and brake data in their raw and converted forms can lead to high accuracy. Therefore, we consider speed, RPM, and brake data to be the most essential features, and the use of raw data alone is considered to be sufficient to achieve the highest accuracy.

CAN provides extensive operational data, but not all features are directly relevant to aggressive driving detection, and some features that could indirectly help reduce false positives are often overlooked. For example, gear position and ultrasonic sensor data, which can minimize the occurrence of false positives in scenarios like parking, where abrupt braking is common, have not been frequently considered in recent studies; existing works [12], [21], [22], [23] often ignore abrupt braking behavior. By contrast, our current study includes gear position and ultrasonic sensor data to enhance the classification of aggressive driving.

B. In-vehicle Network Security Threats and Intrusion Detection Strategies

Due to the CAN protocol's lack of authentication and encryption [29], injection-based attacks have emerged as a critical focus in automotive cybersecurity. Various studies have categorized these attacks into several types. Denial-of-Service (DoS) attacks (see Fig. 2a) involve flooding the CAN bus with high-priority frames, such as ID 0x000, obstructing the delivery of critical messages from other ECUs [30]. Fuzzing attacks (see Fig. 2b) seek to reveal system vulnerabilities by injecting malformed or random CAN messages [30], often causing unexpected behavior or system crashes. Replay attacks (see Fig. 2c) involve capturing and retransmitting previously recorded legitimate messages to illegitimately and repeatedly trigger vehicle functions [30]. To address these

threats, recent study has IDS that utilizes techniques such as Recurrence Quantification Analysis (RQA) with sliding windows to identify anomalies in deterministic patterns [31] and message-sequence graph modeling to detect deviations in communication flows. These advancements highlight the crucial need for real-time anomaly detection and effective message authentication mechanisms to secure contemporary in-vehicle networks.

(R3: 1) Recent advancements in IVN security have introduced various intrusion detection strategies. ECF-IDS [32] combines an Enhanced Cuckoo Filter for rapid screening with a BERT model to analyze CAN IDs, DLC, and payload embeddings, while CANnolo [33] utilizes an unsupervised LSTM autoencoder on READ-preprocessed sequences to detect anomalies via Mahalanobis-distance reconstruction error. Addressing spatial-temporal features, Zhang et al. [34] proposed LiPar, which processes CAN frames as 3D tensors using depthwise-CNN and parallel LSTM branches; however, its resource-adaptation mechanism incurs an approximately 58 ms overhead per cycle. Sun et al. [35] developed KG-ID, a knowledge-graph-based IDS achieving 0.031 ms latency by applying ontology-driven constraints to both frame features and DBC-decoded signal semantics. Despite these advancements, existing designs suffer from high computational complexity and limited generalization due to static training setups. While overt attacks like DoS and fuzzing are detectable, behavior-aware situational replay attacks—particularly those mimicking aggressive driving—remain difficult to distinguish from legitimate driver actions. This gap underscores the need for a unified, lightweight framework capable of differentiating driver-induced aggressiveness from malicious cyber-attacks in real-time.

C. Lightweight Convolutional Models for Resource-Constrained Devices

() Deep learning has been shown to be more effective than traditional methods in extracting complex features from large datasets, thus leading to better accuracies [36]. Although a number of studies have focused on machine learning [7], [8], [9], others have used more advanced approaches [12], [22], [23], [24]. However, a more efficient approach could be achieved by using a simplified algorithm that utilizes specific, correlated functional inputs, thus reducing complexity while maintaining performance. CNNs are particularly useful for analyzing chronological data as it can capture local dependencies through their convolutional layers.

In the domain of efficient architectures, MobileNetV1 [37] established a benchmark for lightweight design, utilizing depthwise separable convolutions to significantly reduce parameter counts and computational costs compared to standard convolutions. Drawing inspiration from these efficiency principles, we propose a streamlined architecture adapted for 1D vehicular data. Unlike MobileNet's original design intended for high-dimensional RGB images, our approach simplifies the feature extraction process by focusing on pointwise convolutions to handle the structured nature of CAN messages. This design philosophy balances accuracy and performance, making the proposed L-CNN highly suitable for real-time implementation on resource-constrained devices in vehicular environments.

III. Data Collection and Features Extraction

A. Data Collection of Aggressive Driving on IVN

This study adopts the *ECU Direct Approach (EDA)* for in-vehicle data acquisition by directly interfacing with the internal gateway via line-tapping tools [38]. This enables access to raw CAN traffic through the Integrated Control Unit (ICU), following the methodology in [17]. The CAN networks of both vehicles consist of multiple functional IDs from chassis CAN Fig 1. Based on the driver's subjective assessment of aggressive behavior, certain data samples did not reflect aggressive patterns in the payload and were therefore excluded from the training set.

(R1, R3: (2), (3)) Study [39] reports the highest prevalence of aggressive driving among young males aged 22–30 (59%); accordingly, this demographic was targeted during recruitment to ensure sufficient representation of high-risk behavior. As shown in Table II, the study involved 20 participants (16 males, 4 females) from four countries, primarily within the 20–30 age range. Data collection was conducted in two stages: first, a mechanically Restricted vehicle with full driver authority was used to establish a reproducible acquisition protocol (Fig. 4), which was then applied to a Tesla vehicle in the second stage, involving six drivers (two females) aged 20–30 across urban, city, and highway environments (Fig. 3); however, Tesla drivers did not retain full manual control due to the default activation of Collision Avoidance Assist (CAA) [40]. Drivers were not provided with any predefined "aggressive intensity" labels and were instructed only through predefined scenarios, allowing driving intensity to reflect natural behavior and subjective perception. In total, data from 16 Kia drivers and 6 Tesla drivers were selected for model training.

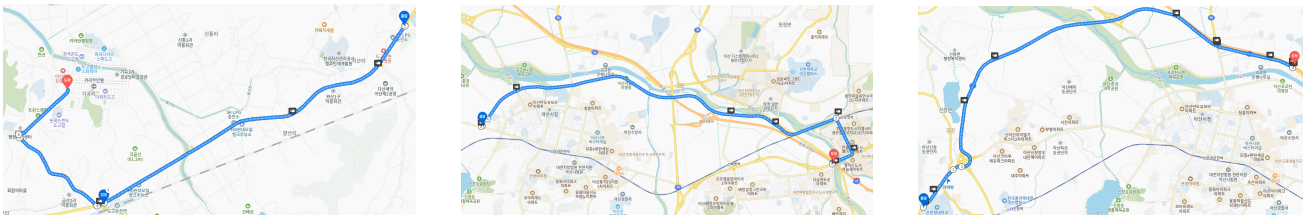
For the Kia driving scenario, aggressive driving is categorized into Safe & Attack-Free (S&AF), Abrupt Brake (AB), Abrupt Lane Change (AL), and Abrupt Acceleration (AA), as summarized in Table III; data were collected under restricted conditions including straight roads, slopes, circular tracks, and turning maneuvers, while the real-world environment reflects milder driving to avoid safety-critical events. For the Tesla driving scenario, driving behavior is divided into *human driving* and *autonomous driving*, where manual driving follows the same aggressive behavior classes as Kia, whereas autonomous driving contains only normal behavior because safety control mechanisms remain enabled by default, preventing aggressive maneuvers and rendering such patterns infeasible to collect; the resulting configuration is reported in Table IV.

B. Data Collection for Cyber-Attacks on IVN

This study focuses on safe driving and cybersecurity measures while using the same features to evaluate both aspects. It considers various types of attacks illustraded in Fig. 2, including DoS, fuzzing, replay, and spoofing attacks, by conducting unauthorized message injections.

TABLE II: (R1, R3: (1), (2) Participant Demographics and Data Collection Details

Driver	Nationality	Age	Gender	Route	Kia	Tesla	Model Training
Driver 1	Uzbekistan	23	Male	Restricted	✓	✗	✓
Driver 2	Korean	24	Male	Restricted	✓	✓	✓
Driver 3	Korean	26	Male	Restricted	✓	✓	✓
Driver 4	Uzbekistan	21	Male	Highway	✓	✗	✓
Driver 5	Korean	20	Male	Urban	✓	✗	✓
Driver 6	Mongolian	20	Male	City	✓	✗	✓
Driver 7	Korean	35	Female	Highway	✓	✗	✗
Driver 8	Korean	50	Male	Urban	✓	✗	✓
Driver 9	Korean	30	Female	Highway	✓	✗	✗
Driver 10	Korean	30	Male	Restricted	✓	✓	✓
Driver 11	Korean	35	Male	Restricted	✓	✓	✓
Driver 12	Mongolian	24	Male	City	✓	✗	✓
Driver 13	Uzbekistan	23	Male	Restricted	✓	✗	✓
Driver 14	Uzbekistan	26	Male	Restricted	✓	✗	✓
Driver 15	Uzbekistan	26	Male	Restricted	✓	✗	✓
Driver 16	Mongolian	22	Male	Restricted	✓	✗	✓
Driver 17	Iranian	36	Male	Highway	✓	✗	✗
Driver 18	Korean	23	Female	Restricted	✓	✓	✓
Driver 19	Korean	23	Female	Restricted	✓	✓	✓
Driver 20	Korean	25	Male	Highway	✓	✗	✗



(a) Urban (b) City (c) Highway

Fig. 3: Data collection route in the real-world environment to gather aggressive driving patterns, including safe driving, abrupt braking, lane changes, and acceleration.

TABLE III: (R3:2) Kia Dataset Composition by Driving Behavior and Attack Type

Group	Class	Num of CAN Frames	Duration (Sec)
Manual Driving	S&AF	17 826 000	9 000
	AB	1 415 000	700
	AL	1 798 000	900
	AA	1 349 000	700
Conventional Attacks	DoS	16 000	80
	Fuzz	140 000	150
	Replay	155 000	150
Behavior-aware Situational Replay	AAB	100 000	180
	AAL	100 000	180
	AAA	100 000	180

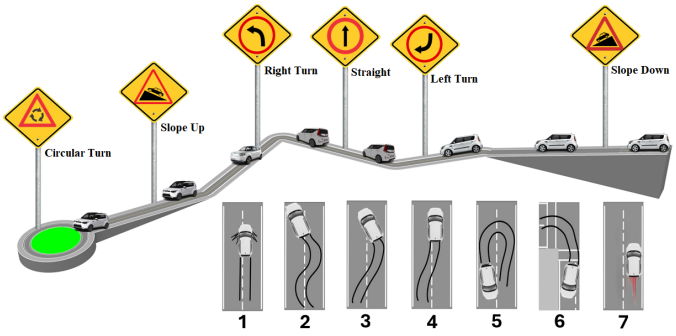


Fig. 4: Data collection route in the restricted environment to gather aggressive driving patterns, including safe driving, abrupt braking (1), lane-change (2 to 6) and acceleration (7).

- The DoS attack (Fig. 2a) focused on CAN IDs ”0x000” because these IDs had the highest priority in the target vehicle.
- Fuzzing attacks (Fig. 2b) targeted a wide range of IDs, from ”0x000” to ”0x1000,” covering both high and low-priority IDs.
- Replay attacks (Fig. 2c) were conducted in two phases:
  - First, normal driving data was periodically injected.
  - *Behavior-aware situational replay injection attacks* were designed to simulate context-dependent threats, including abrupt braking, lane deviation, and acceleration manipulation, corresponding to AAB, AAL, and AAA as listed in Table III and Table IV. The injection intervals for all attack scenarios ranged

TABLE IV: (R3:2) Tesla Dataset Composition by Driving Mode and Attack Type

Mode	Class	Num of CAN Frames	Duration (Sec)
Manual Driving	S&AF	600 000	200
	AB	220 000	75
	AL	900 000	300
	AA	164 000	60
Conventional Attacks	DoS	100 000	200
	Fuzz	100 000	200
	Replay	100 000	200
Behavior-aware Situational Replay	AAB	91 000	200
	AAL	91 000	200
	AAA	91 000	200
Autonomous Driving	S&AF	820 000	270
Autonomous Conventional Attacks	DoS	50 000	135
	Fuzz	61 500	135
	Replay	50 000	135
Autonomous Behavior-aware Situational Replay	AAB	37 000	135
	AAL	14 000	135
	AAA	40 000	135

between 0.1 and 0.5 seconds.

C. Features Extraction and Data Analysis

(R1, R3: (2), (3)) To identify which CAN IDs control specific vehicle functions, a trace-guided fuzzing methodology was employed on the captured in-vehicle CAN traffic. First, a baseline was established by analyzing CAN traces collected during normal vehicle operation through the ECU direct approach. This baseline cataloged all active CAN IDs, their transmission frequencies, and typical payload patterns observed during standard driving operations.

The captured CAN IDs were then classified into two categories based on their payload characteristics:

- **Limited-payload IDs**: CAN messages with values constrained to narrow, discrete ranges corresponding to specific vehicle states (e.g., gear position)
- **Arbitrary-payload IDs**: CAN messages with values spanning broad or continuous ranges reflecting dynamic sensor readings (e.g., engine RPM, vehicle speed)

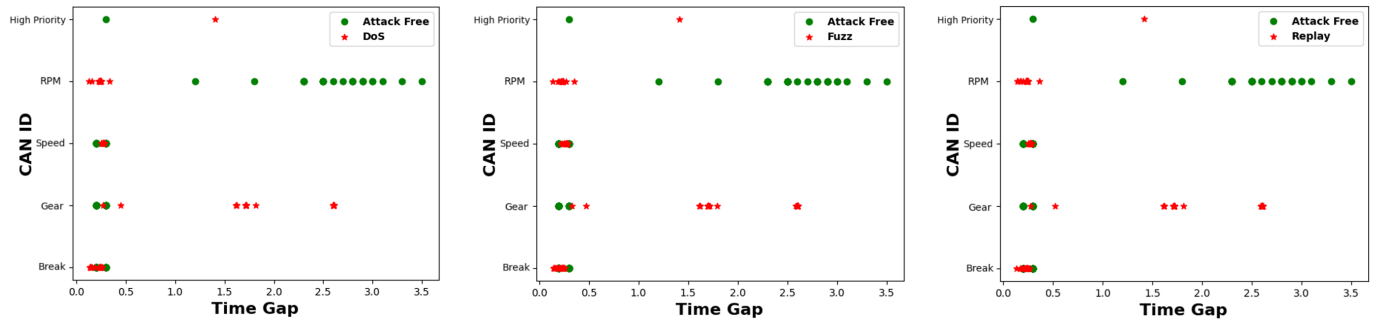
For each identified CAN ID, fuzzing test cases were generated by systematically mutating specific data bytes while maintaining the original message structure. For limited-payload IDs, all discrete values within and slightly beyond the observed operational range were exhaustively tested. For arbitrary-payload IDs, boundary value analysis and stratified sampling across the payload space were applied. Each fuzzed message was injected onto the CAN bus through the same ICU access point.

During injection, the vehicle was monitored for physical responses including changes in instrument cluster displays, activation of warning indicators, gear shifts, actuator movements, and any abnormal vehicle behavior. Each observed response was validated through multiple repetition cycles to confirm reproducibility and establish causal relationships between injected payloads and physical effects. By correlating specific payload mutations with consistent physical responses, the functional semantics of each CAN ID were reverse-engineered and a mapping between CAN message structures and their corresponding vehicle subsystems was constructed.

Minimal feature extraction can optimize overall accuracy by reducing the operational flow and computational costs. Speed, RPM, brakes, and gear exhibit identical data generation frequencies. Due to their identical data generation frequencies, they also contribute to identifying cyberattacks. Speed and RPM are interconnected through the vehicle's transmission system, with lower gears requiring higher RPM to maintain the same speed as required with higher gears. When brakes are applied, both speed and RPM typically decrease as a result of reduced engine load. Gear position directly affects the RPM-speed relationship, as lower gears result in higher RPM at lower speeds. Various driving actions, such as accelerating, braking, and lane changes, are reflected in the payload data of specific CAN IDs, which include parameters like speed, RPM, brake status, and gear position. These values fluctuate continuously in response to both the driver's actions and vehicle dynamics.

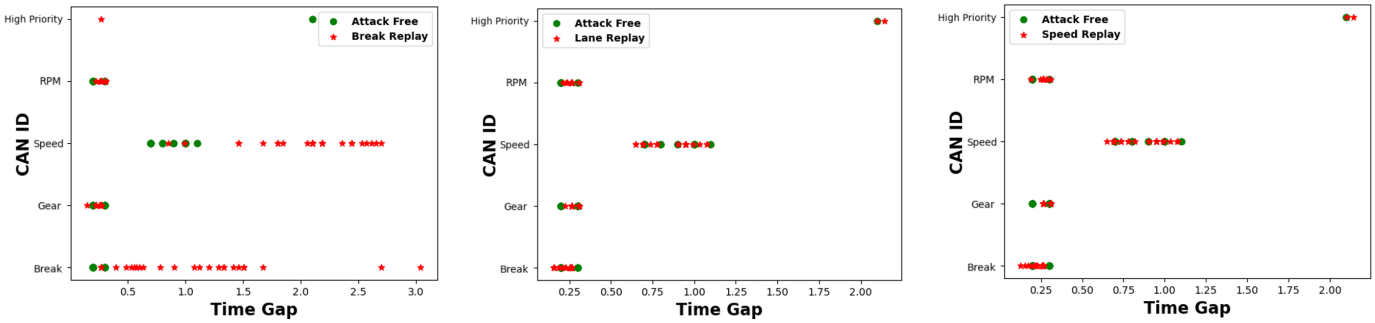
(R1: 1, 11) For the cross-driver and cross-scenario analysis of Kia, during lane changes, the gear byte at position eight consistently maintains a stable value of "00," while other acceleration and break byte values fluctuate arbitrarily. In contrast, during acceleration, byte position six, corresponding to RPM, exhibits arbitrary variations, whereas the remaining byte values stay within a fixed range. Similarly, during both abrupt lane changes and acceleration, brake-related values appear irregularly due to the common use of braking in both scenarios for vehicle control. As a result, brake signals are observed in both lane change and acceleration states. Compared with normal driving, byte position eight in the brake frequently shows "0F" or "10" across all drivers. For Tesla, during acceleration, byte position six in the speed frequently shows values "A0" and "A8," while in the gear, "86" is commonly observed under abrupt acceleration and transitions to "88" during abrupt lane changes and braking.

Cyberattacks that inject unauthorized CAN messages disrupt the natural sequencing of functional CAN IDs and their characteristic message timing patterns. Under normal conditions, the behavioral sequence of Brake, Gear, Speed, and RPM IDs follows a stable temporal order, and the periodicity of their associated time gaps remains consistent. The high-priority multifunction ID, generated automatically every 0.2 seconds on Kia and every 0.16 seconds on Tesla and triggered immediately during specific driver actions, serves as a highly timing-sensitive reference signal. Because their emission pattern is strictly periodic, even small deviations in their arrival intervals provide a reliable indicator of threat attempts, or timing manipulation. (R3:4) Thus, high-priority multi function ID directly support intrusion detection, signal that exposes abnormal message injection. Ultrasonic sensor readings are incorporated to provide contextual awareness during low-speed maneuvers—such as parking, reversing, or obstacle avoidance—where sudden braking or irregular motion is normal and should not be interpreted as aggressive behavior. By distinguishing legitimate low-speed braking events from suspicious



(a) DoS (b) Fuzz (c) Replay

Fig. 5: Physical differences between attack-free and attack scenarios on Speed, RPM, Brake, Gear, and high-priority IDs, with data injected at varying frequencies.



(a) Abrupt Break Replay (b) Abrupt Lane Change Replay (c) Abrupt Acceleration Replay

Fig. 6: Situational Replay Attack: Aggressive driving data injection by matching source frequency on IVN.

ones, the ultrasonic sensor helps significantly reduce false positives, improving both the reliability and robustness of the detection system. In Fig. 5 and Fig. 6, the x-axis represents the message time gap and the y-axis represents the corresponding CAN ID. Under attack-free conditions (green), periodicity and ID ordering remain stable; however, injected attacks (red) disrupt both properties. Similarly, Fig. 6a, Fig. 6b, and Fig. 6c illustrate behavior-aware situational replay scenarios in which injected brake, lane-change, or speed-related messages mimic legitimate frequencies yet still break the natural temporal dependencies. While abrupt braking produces large detectable deviations, lane-change and speed-manipulation attacks cause more subtle disruptions. The combination of timing-sensitive high-priority multi function ID and contextual Ultrasonic Sensor data enables the proposed model to detect both prominent and subtle abnormalities in live CAN streams with high accuracy.

IV. Implementation of Experimental Methods

(R2:4, R3:5) In this study, data preprocessing was intentionally simplified to achieve high accuracy with low computational overhead, as shown in Algorithm 1. The system operates as a continuous real-time pipeline that receives live CAN frames, filters only six functional filtered behavior-related IDs and computes the time interval between consecutive messages. Each selected frame is label-encoded to treat hexadecimal values as categorical data and then normalized using Min–Max scaling. A rolling buffer accumulates incoming messages until a sliding window is formed, after which the window is reshaped into a $(1 \times \text{chunk\_size} \times D)$ tensor for L-CNN inference. Through extensive testing, a chunk size of 20 with 50% overlap (approximately 20 ms) was identified as the optimal configuration, providing near real-time response while preserving high accuracy for both safety (aggressive-driving detection) and security (attack-aware behavior modeling).

TABLE V: R2:(C 4) Parameter Efficiency: L-CNN vs. Existing Architectures

Architecture	Parameters	vs. L-CNN
MobileNetV1 (original, $\alpha = 1.0$)	4.2M	1260× larger
DriverBehaviorCNN [22] (2×32, 7×7)	107K	32× larger
GRU (2×64)	26.9K	8× larger
L-CNN (proposed)	**3.3K**	**baseline**

The proposed L-CNN design in Fig. 7 is intentionally lightweight, featuring progressive channel expansion ($8 \rightarrow 16 \rightarrow 32 \rightarrow 64$) combined with global average pooling that enables compact feature aggregation with strong generalization, reducing parameter redundancy and overfitting compared to deeper or fully connected alternatives. Unlike image-based applications where depthwise separable convolutions exploit spatial locality, our input comprises structured CAN message sequences where cross-channel relationships are more critical than spatial patterns. Therefore, we adopt pointwise (1×1) convolutions after the initial standard convolution, which efficiently mix channel-wise features while maintaining low computational cost—particularly beneficial

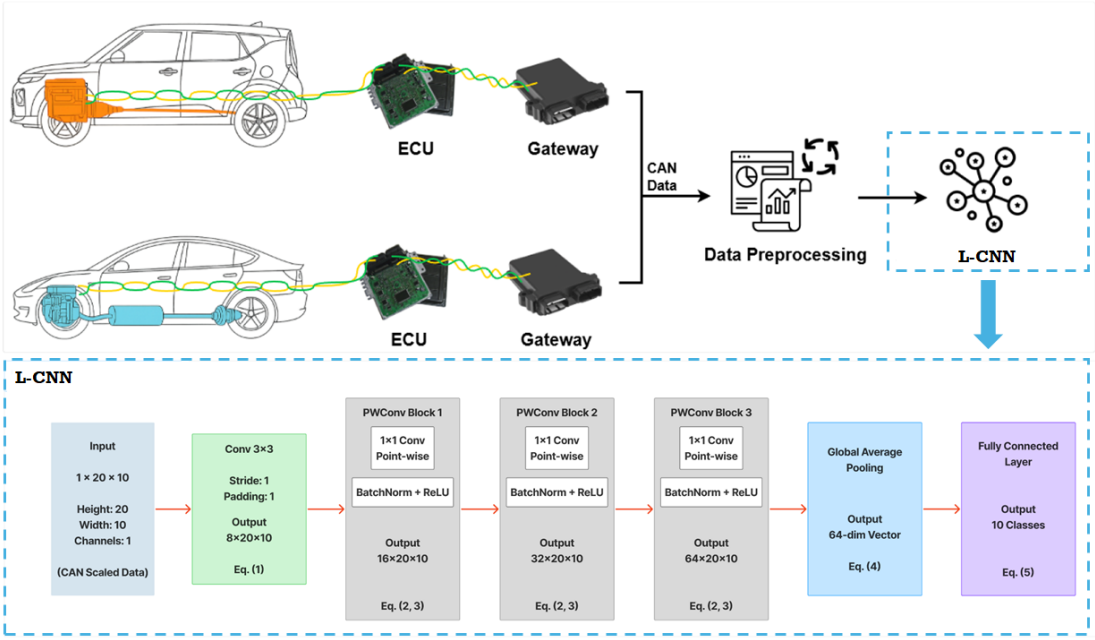


Fig. 7: (R1:3) Intuitive illustration of the proposed model's end-to-end data flow. The upper portion presents the high-level pipeline from CAN signal acquisition (ECU → Gateway → Preprocessing) to L-CNN inference and final driving-behavior classification. The lower portion provides the low-level tensor transformation path, showing how the 1×20×10 input tensor is progressively mapped through convolution, pointwise-convolution blocks, and global average pooling to produce a 64-dimensional feature vector and the final ten output classes.

given the small channel dimensions (8 → 64) throughout the network. This architecture preserves channel-wise discriminative features while minimizing computational overhead, making it suitable for real-time IVN deployment on resource-constrained embedded devices. As quantified in Table V, the proposed L-CNN achieves orders-of-magnitude parameter reduction compared to conventional deep architectures, confirming that high detection accuracy is attained with substantially lower computational cost. Once inference is performed, the predicted driving-behavior label is immediately forwarded to downstream modules such as logging services, visualization dashboards, or ECU-level alert systems, enabling continuous on-vehicle monitoring with minimal latency.

The model begins with applying a 2D convolution to the input tensor of size $1 \times 20 \times 10$, where the height (20) represents rows of features, the width (10) corresponds to columns (including CAN ID, payloads, and time gap), and the channel is 1 due to the tabular nature of the data, similar to a grayscale image. The initial convolutional layer uses a 3×3 kernel, stride of 1, and padding of 1, ultimately producing an output tensor of size $8 \times 20 \times 10$. This operation can be expressed as follows:

$$X_{1,i,j,m} = \sum_{p=0}^{2} \sum_{q=0}^{2} X_{\text{input},i+p-1,j+q-1,1} \times W_{p,q,1,m} + b_m \quad (1)$$

In Equation: 1, $X_{\text{input}} \in \mathbb{R}^{1\times20\times10}$ represents the input tensor with shape $1 \times 20 \times 10$. The convolutional kernel $W \in \mathbb{R}^{3\times3\times1\times8}$ has a shape of $3 \times 3 \times 1 \times 8$, where the first two dimensions (3, 3) represent the height and width of the kernel, the third dimension (1) represents the number of input channels, and the fourth dimension (8) represents the number of output filters. The bias term b_m corresponds to the m-th output filter, and the resulting output tensor $X_1 \in \mathbb{R}^{8\times20\times10}$. Here, i and j are spatial indices representing positions along the height and width of the output tensor, respectively, m is the index representing the output filter, and p and q are indices representing positions within the convolutional kernel.

Next, the model includes three convolutional blocks, each using a pointwise convolution (a 1×1 convolution) to adjust the number of channels without changing the spatial dimensions. Each pointwise convolution block is followed by batch normalization and ReLU activation to both enhance training stability and introduce non-linearity. The pointwise convolution operation for these layers can be generalized as follows:

$$Y_{i,j,k'} = \sum_{k}^{D_{in}} X_{i,j,k} \times W_{1,1,k,k'} + b_{k'} \quad (2)$$

$$Y_{i,j,k'} = X_{i,j,1} \times W_{1,1,1,k'} + b_{k'} \quad (3)$$

In Equation: 2, $Y_{i,j,k'}$ represents the output at position (i, j) in the k'-th channel, $X_{i,j,k}$ is the input at position (i, j) in the k-th channel, and $W_{1,1,k,k'}$ denotes the 1×1 convolutional filter. D_{in} is crucial here as it defines how many input channels are considered in the summation for each output channel; here the number of channels is 1. Equation: 3 represents the simplification. This equation is applied in each of the three blocks, with varying input and output channels:

Algorithm 1: (R3:5) End-to-End Deployment

Live CAN stream S_{CAN}, trained artifacts $\{\texttt{Encoders}, \texttt{Scaler}, \texttt{L\_CNN.pth}\}$, $chunk\_size$, $overlap$ Real-time predicted labels and alerts
Phase 0: Service Initialization
Load pre-trained label encoder E_{label} and scaler S_{minmax}
Load L-CNN model from `L_CNN.pth`
Set model to `eval()` and disable gradients
Initialize empty buffer df_{buf}
$step \leftarrow chunk\_size \times (1 - overlap)$
Phase 1: Inference Loop
while *service is running* **do**
  Read new batch B from S_{CAN}
  Append B to df_{buf}
  Step 1: Feature Extraction
  df_{buf}[Time_Delta] $\leftarrow df_{\text{buf}}$[Time_Offset].diff(1)
  $df_{\text{buf}} \leftarrow df_{\text{buf}}[df_{\text{buf}}[\text{CAN\_ID}] \in IDs_{Filtered}]$
  $df_{\text{buf}} \leftarrow df_{\text{buf}}[\{\text{CAN\_ID}, P_{1:8}, \text{Time\_Delta}\}]$
  if $|df_{buf}| \geq chunk\_size$ **then**
    Step 2: Preprocessing
    Let D be number of columns
    for $j \in \{1, \ldots, D\}$ **do**
      $df_{\text{buf}}^{(j)} \leftarrow E_{label}.transform(df_{\text{buf}}^{(j)})$
    $df_{\text{buf}} \leftarrow S_{minmax}.transform(df_{\text{buf}})$
    Step 3: Sliding Window
    $S \leftarrow df_{\text{buf}}[|df_{\text{buf}}| - chunk\_size : |df_{\text{buf}}|]$
    $X \leftarrow reshape(S, (1, chunk\_size, D))$
    Step 4: Inference
    $output \leftarrow model(X)$
    $\hat{y} \leftarrow \arg\max(output)$
    Step 5: Output
    Emit $\hat{y}$ to logger / dashboard / ECU alerts
    Optionally store $(S, \hat{y})$

- **First Block:** The input tensor $X_1 \in \mathbb{R}^{8\times20\times10}$ undergoes a pointwise convolution followed by batch normalization and ReLU activation to produce an output tensor $X_2 \in \mathbb{R}^{16\times20\times10}$.
- **Second Block:** The resulting $X_2 \in \mathbb{R}^{16\times20\times10}$ is further processed by a pointwise convolution, batch normalization, and ReLU activation, eventually yielding $X_3 \in \mathbb{R}^{32\times20\times10}$.
- **Third Block:** Finally, $X_3 \in \mathbb{R}^{32\times20\times10}$ undergoes another pointwise convolution followed by batch normalization and ReLU activation, resulting in $X_4 \in \mathbb{R}^{64\times20\times10}$.

After these convolutional blocks, the model applies a global average pooling layer to the $64 \times 20 \times 10$ feature map, which condenses each feature map into a single value, ultimately producing the following 64-dimensional vector:

$$X_{\text{pool}} = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X_{4,i,j} \tag{4}$$

In Equation: 4, $H = 20$ is the height of the feature map, $W = 10$ is the width of the feature map, $X_{4,i,j}$ represents the input feature map at position (i, j) of the 4th layer, the term $\frac{1}{H\times W}$ represents the average pooling operation, where the sum of all values in the channel is divided by the total number of elements in that channel, and i and j are indices that run over the height and width of the feature map.

Finally, this 64-dimensional vector is passed through a fully connected layer to produce the following class scores for the five output classes:

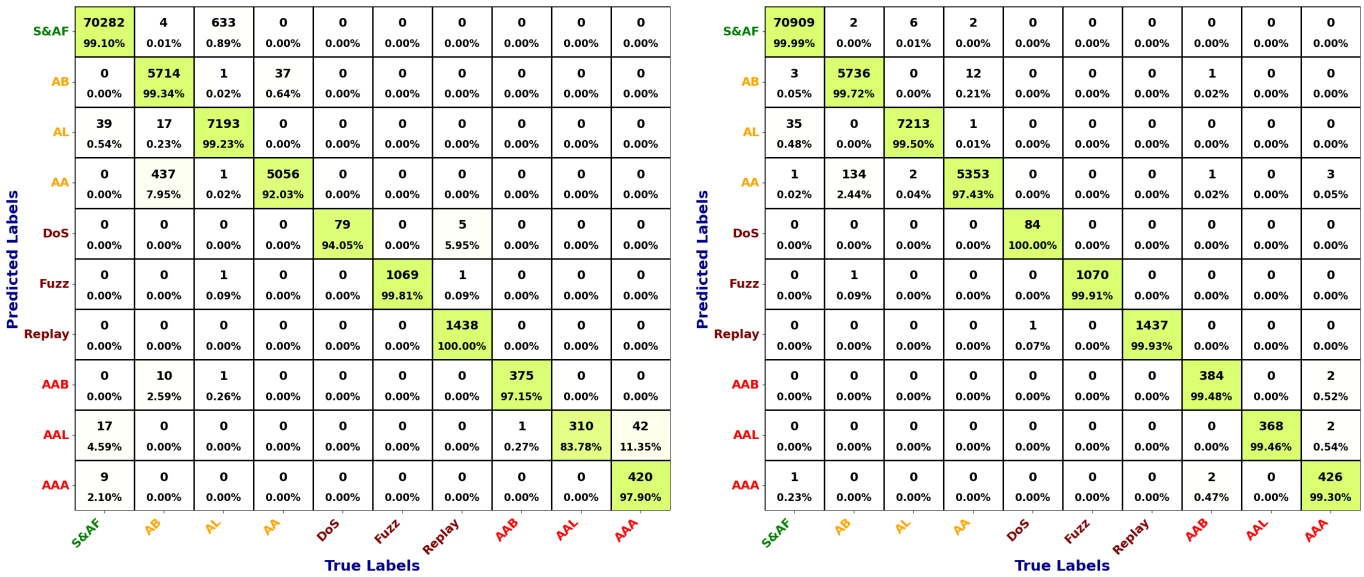
$$y_c = \sum_{q=1}^{64} X_{\text{pool}} \times W_{\text{fc},q,c} + b_{\text{fc},c} \tag{5}$$

In Equation: 5, W_{fc} is the weight matrix of the fully connected layer, b_{fc} is the bias term, and y_c represents the output score for the c-th class. The bias helps the model to better fit the data by providing an offset that can shift the activation function, thus allowing for more flexibility in the learning process. This architecture effectively balances computational efficiency and performance by using pointwise convolutions to adjust channel dimensions, which ultimately makes it suitable for devices with constrained computational resources.

V. Experimental Evaluation

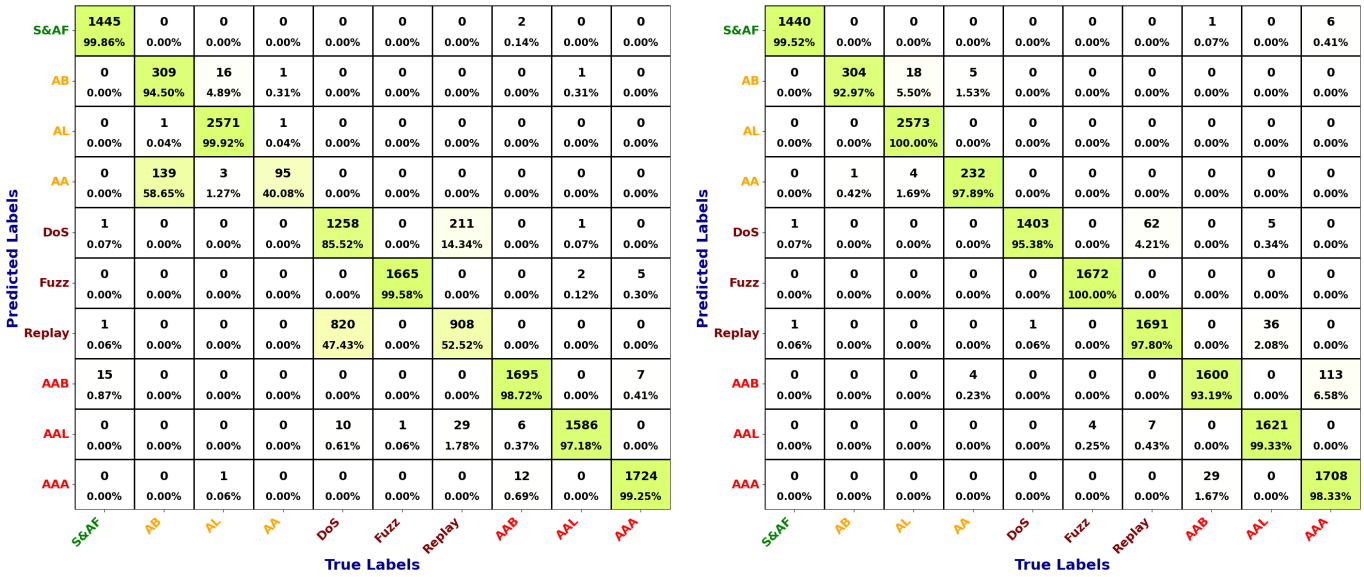
(R2:4) Following standard evaluation practice precision, recall, and F1-score were adopted as performance metrics [41], and computed for each driving behavior and attack class across three datasets: Kia manual driving, Tesla manual driving, and Tesla autonomous driving. The detailed class-wise results include 95% confidence intervals [42]. Both GRU and L-CNN architectures were used for the detection mechanism. The GRU model architecture follows our prior optimized IDS design [30] for CAN time-series learning, where extensive hyperparameter exploration across multiple platforms confirmed GRU as the most effective RNN family member for in-vehicle attack detection under resource constraints. The optimized GRU configuration consists of 2 layers with 64 hidden units, 0.2 dropout rate, and processes input sequences of shape $(batch, 20, 10)$ through a final fully connected layer with softmax activation, trained using NAdam optimizer with learning rate 0.001 and CrossEntropyLoss. In addition, we benchmarked L-CNN against a CNN baseline [22] on Raspberry Pi 4 to evaluate deployment feasibility by comparing preprocessing cost, inference latency, total detection time, and memory usage, which confirms L-CNN as the more suitable architecture for real-time edge deployment.

The algorithm was evaluated in two steps to balance accuracy and detection time. Initially, various chunk sizes were tested, starting with 30 samples, which was identified as the minimum chunk size with a 50% overlap based on recent studies. Subsequently, chunk sizes of 20, 10, and 5 samples were applied, resulting in detection times of 60ms, 40ms, 20ms, and 10ms respectively, with the highest accuracy observed at 40ms without overlapping. To further reduce the detection time, three overlap parameters were tested: 25%, 50%, and 75%. A 50% overlap with a chunk size of 20 samples yielded the optimal detection time of 20ms along



(a) GRU Kia (b) L-CNN Kia

Fig. 8: (R1, R2):(1, 6, 7, 8, 11), (4, 6) Kia Manual Driving Dataset Evaluating Aggressive Driving (AB, AL, AA), DoS, Fuzzing, Replay, and Behavior-aware Situational Replay Attacks (AAB, AAL, AAA)

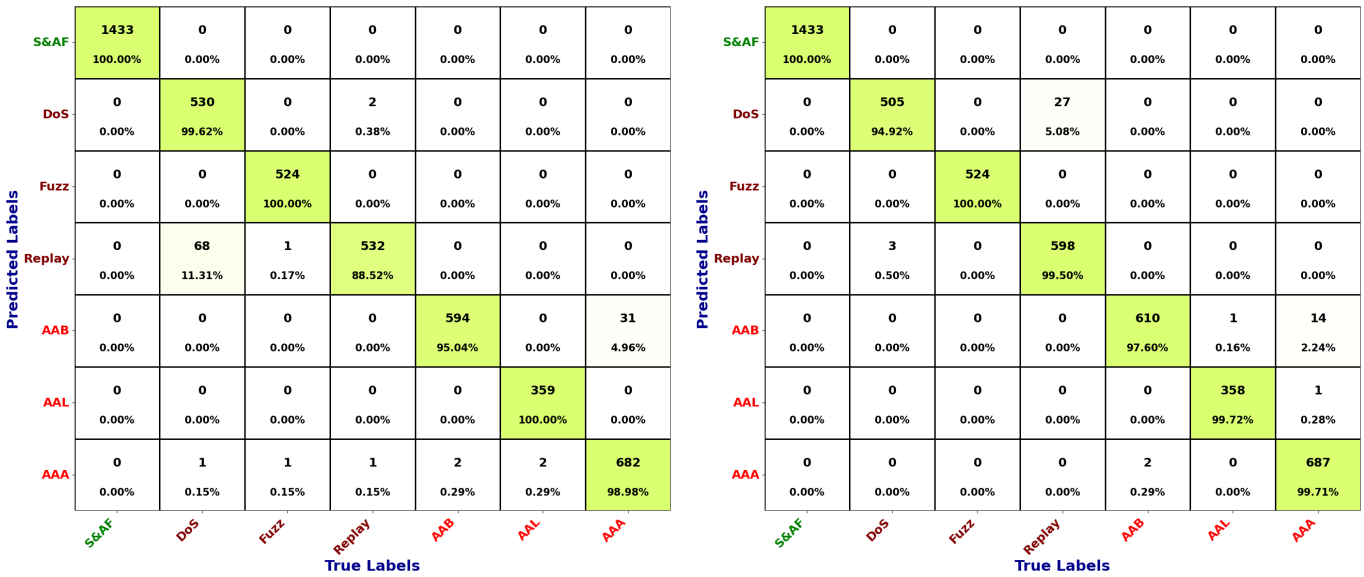


(a) GRU Tesla (b) L-CNN Tesla

Fig. 9: (R1, R2):(1, 6, 7, 8, 11), (4, 6) Tesla Manual Driving Dataset Evaluating Aggressive Driving (AB, AL, AA), DoS, Fuzzing, Replay, and Behavior-aware Situational Replay Attacks (AAB, AAL, AAA)

with the highest accuracy. Notably, the use of smaller chunk sizes introduced challenges, including insufficient information for accurate prediction on dynamic situation and an increased algorithm activation rate, ultimately leading to computational overflow.

In Fig. 8 reveals distinct performance patterns between L-CNN and GRU on the Kia human-driving dataset. For Safe & Attack-Free (S&AF) driving, both models achieve high accuracy (GRU: 99.10% vs L-CNN: 99.99%), though L-CNN shows slightly better performance with only 10 misclassifications compared to GRU's 637. However, critical differences emerge in attack detection. For Abrupt Acceleration (AA), L-CNN achieves 97.43% accuracy while GRU drops to 92.03%, misclassifying 437 AA samples as AB—indicating confusion between acceleration patterns. In DoS detection, L-CNN maintains perfect 100.00% accuracy whereas GRU achieves 94.05%, misclassifying 5 instances as Replay. For behavior-aware situational replay attacks, L-CNN demonstrates superior performance: AAB (99.48% vs 97.15%), AAL (99.46% vs 83.78%), and AAA (99.30% vs 97.90%). GRU's AAL detection suffers significantly with only 83.78% recall, misclassifying 17 samples as S&AF and 42 as AAA—suggesting



(a) GRU Tesla

(b) L-CNN Tesla

Fig. 10: (R1, R2):(1, 6, 7, 8, 11), (4, 6) Tesla Autonomous Driving Dataset Evaluating DoS, Fuzzing, Replay, and Behavior-aware Situational Replay Attacks (AAB, AAL, AAA)

vulnerability to lane-change attack patterns. Both models excel in Fuzz and Replay detection ($> 99.8\%$).

In Fig. 9 shows that L-CNN outperforms GRU across most classes in the Tesla human-driving dataset. For Safe & Attack-Free (S&AF) driving, both models perform strongly, with L-CNN achieving 99.52% and GRU 99.86%. For aggressive maneuvers, L-CNN demonstrates superior performance: Abrupt Braking (AB) reaches 92.97% versus GRU's 94.50%; Abrupt Lane Change (AL) achieves perfect 100.00% versus GRU's 99.92%; and Abrupt Acceleration (AA) shows L-CNN at 97.89% versus GRU's catastrophic 40.08%—GRU misclassifies 139 of 237 AA samples (58.65%) as AB, indicating severe confusion between acceleration patterns. For cyber-attacks, L-CNN achieves superior DoS detection at 95.38% versus GRU's 85.52% (GRU misclassifies 211 samples as Replay), and perfect Fuzz detection at 100.00% versus GRU's 99.58%. Replay detection reveals the largest performance gap: L-CNN achieves 97.80% while GRU drops dramatically to 52.52%, misclassifying 820 of 1729 Replay samples as DoS. For behavior-aware situational replay attacks, L-CNN remain consistency label: AAB (93.19%) , AAL (99.33%) , and (AAA 98.33%).

In Fig. 10 shows that both models perform strongly under autonomous driving, with L-CNN exhibiting higher class-wise reliability in most attack categories. Both models achieve perfect recognition for Safe & Attack-Free (S&AF) behavior (100.00%). For conventional attacks, GRU outperforms L-CNN in DoS detection (99.62% vs 94.92%) L-CNN misclassifies 27 DoS samples as Replay, while GRU only misclassifies 2. Both models achieve perfect Fuzz detection (100.00%). Replay detection reveals the largest performance gap: L-CNN achieves 99.50% with only 3 misclassifications, while GRU drops to 88.52%, misclassifying 68 Replay samples (11.31%) as DoS. For behavior-aware situational replay attacks, L-CNN demonstrates remain consistency label: AAB (97.60%), AAL (99.72%) , and AAA (99.71%). GRU misclassifies 31 AAB samples as AAA (4.96%), while L-CNN misclassifies only 14 (2.24%).

Across all three datasets, L-CNN demonstrates substantially greater robustness and cross-dataset stability in critical attack scenarios, indicating strong generalization without overfitting. In Kia manual driving, L-CNN excels in acceleration-based and lane-change injection scenarios where GRU shows vulnerability. In Tesla manual driving, L-CNN significantly outperforms in AA detection and Replay attacks where GRU exhibits severe classification instability. In Tesla autonomous driving, L-CNN shows superior Replay and AAB detection, while GRU achieves slightly better DoS performance and perfect AAL detection. The most critical difference appears in GRU's handling of temporal attack patterns, particularly Replay attacks in manual driving and acceleration pattern recognition, where performance degrades dramatically. L-CNN maintains consistently high performance across all attack types and datasets, suggesting that its spatial feature extraction approach provides more robust and generalizable defense against sophisticated cyber-attack scenarios. This stability across diverse driving conditions and attack vectors demonstrates that L-CNN effectively learns discriminative patterns without memorizing dataset-specific artifacts, a key indicator of successful model generalization rather than overfitting.

The classification results in Table VII reveal that L-CNN consistently outperforms GRU in robustness and stability. For Kia Manual driving, L-CNN achieves superior F1-scores across most attack types, particularly for AAL (0.9973 vs 0.9118) and DoS (0.9941 vs 0.9693). GRU shows notable weakness in AAL detection with only 83.78% recall. In Tesla Manual driving, the performance gap widens: GRU struggles significantly with AA (56.89% F1 vs 97.07%), DoS

TABLE VI: (R1, R2):(1, 9, 11),(4, 6) Comparison of GRU and L-CNN Across Kia, Tesla Manual, and Tesla Autonomous Driving

Dataset	Class	Precision		Recall		F1		CI Lower		CI Upper	
		GRU	L-CNN	GRU	L-CNN	GRU	L-CNN	GRU	L-CNN	GRU	L-CNN
Kia Human	S&AF	0.9991	**0.9994**	0.9910	**0.9999**	0.9950	**0.9996**	0.9903	**0.9997**	0.9917	**0.9999**
	AB	0.9243	**0.9767**	0.9934	**0.9972**	0.9576	**0.9868**	0.9909	**0.9955**	0.9952	**0.9983**
	AL	0.9186	**0.9989**	0.9923	**0.9950**	0.9540	**0.9970**	0.9900	**0.9931**	0.9940	**0.9964**
	AA	0.9927	**0.9972**	0.9203	**0.9743**	0.9551	**0.9856**	0.9128	**0.9698**	0.9271	**0.9782**
	DoS	**1.0000**	0.9882	0.9405	**1.0000**	0.9693	**0.9941**	0.8681	**0.9563**	0.9743	**1.0000**
	Fuzz	**1.0000**	**1.0000**	0.9981	**0.9991**	0.9991	**0.9995**	0.9932	**0.9947**	0.9995	**0.9998**
	Replay	0.9958	**1.0000**	**1.0000**	0.9993	0.9979	**0.9997**	**0.9973**	0.9961	**1.0000**	0.9999
	AAB	**0.9973**	0.9897	0.9715	**0.9948**	0.9843	**0.9922**	0.9497	**0.9813**	0.9840	**0.9986**
	AAL	**1.0000**	**1.0000**	0.8378	**0.9946**	0.9118	**0.9973**	0.7968	**0.9805**	0.8719	**0.9985**
	AAA	0.9091	**0.9838**	0.9790	**0.9930**	0.9428	**0.9884**	0.9606	**0.9796**	0.9889	**0.9976**
Tesla Human	S&AF	0.9884	**0.9986**	**0.9986**	0.9952	0.9935	**0.9969**	**0.9950**	0.9900	**0.9996**	0.9977
	AB	0.6882	**0.9967**	**0.9450**	0.9297	0.7964	**0.9620**	**0.9147**	0.8967	**0.9649**	0.9527
	AL	**0.9923**	0.9915	0.9992	**1.0000**	**0.9957**	**0.9957**	0.9972	**0.9985**	0.9998	**1.0000**
	AA	**0.9794**	0.9627	0.4008	**0.9789**	0.5689	**0.9707**	0.3405	**0.9516**	0.4643	**0.9910**
	DoS	0.6025	**0.9993**	0.8552	**0.9538**	0.7069	**0.9760**	0.8363	**0.9418**	0.8723	**0.9634**
	Fuzz	**0.9994**	0.9976	0.9958	**1.0000**	0.9976	**0.9988**	0.9914	**0.9977**	0.9980	**1.0000**
	Replay	0.7909	**0.9608**	0.5252	**0.9780**	0.6312	**0.9693**	0.5016	**0.9700**	0.5486	**0.9839**
	AAB	**0.9883**	0.9816	**0.9872**	0.9319	**0.9878**	0.9561	**0.9807**	0.9189	**0.9915**	0.9428
	AAL	**0.9975**	0.9753	0.9718	**0.9933**	**0.9845**	0.9842	0.9626	**0.9880**	0.9788	**0.9962**
	AAA	**0.9931**	0.9349	**0.9925**	0.9833	**0.9928**	0.9585	**0.9872**	0.9761	**0.9956**	0.9884
Tesla Autonomous	S&AF	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.9973**	**0.9973**	**1.0000**	**1.0000**
	DoS	0.8848	**0.9941**	**0.9962**	0.9492	0.9372	**0.9712**	**0.9864**	0.9272	**0.9990**	0.9649
	Fuzz	0.9962	**1.0000**	**1.0000**	**1.0000**	0.9981	**1.0000**	**0.9927**	**0.9927**	**1.0000**	**1.0000**
	Replay	**0.9944**	0.9568	0.8852	**0.9950**	0.9366	**0.9755**	0.8572	**0.9854**	0.9083	**0.9983**
	AAB	**0.9966**	0.9967	0.9504	**0.9760**	0.9730	**0.9863**	0.9305	**0.9608**	0.9648	**0.9854**
	AAL	0.9945	**0.9972**	**1.0000**	0.9972	**0.9972**	**0.9972**	**0.9894**	0.9844	**1.0000**	0.9995
	AAA	0.9565	**0.9786**	0.9898	**0.9971**	0.9729	**0.9878**	0.9792	**0.9895**	0.9951	**0.9992**

(70.69% vs 97.60%), and Replay (63.12% vs 96.93%), while L-CNN maintains consistently high scores above 0.95 for all classes. Both models excel in Tesla Autonomous scenarios with near-perfect detection, though L-CNN edges ahead in DoS (97.12% F1 vs 93.72%) and Replay (97.55% vs 93.66%). L-CNN demonstrates exceptional consistency with 24 out of 27 classes achieving Recall above 0.95, compared to GRU's 19 classes. The confidence intervals further confirm L-CNN's stability, showing tighter bounds across all metrics. Overall, L-CNN proves more robust to lane-change attacks, replay scenarios, and DoS conditions, while GRU exhibits dataset-specific vulnerabilities that limit its generalization capability.

TABLE VII: (R1: 4) Detection time DriverBehaviorCNN vs. L-CNN on Raspberry Pi 4

Metric	DriverBehaviorCNN	Modified L-CNN
Input Features	Acceleration and Gravity signals from Smartphone and CAN	Only CAN
Preprocessing	Recurrence-plot images	Raw tensor
Chunk (ms)	1	20
Preprocess (ms)	171.05	29.18
Inference (ms)	173.63	43.20
Total (ms)	339.66	92.38
Memory (KB)	10226.16	112.45

The comparative analysis in Table VII highlights the performance differences between the DriverBehaviorCNN and L-CNN models on a Raspberry Pi 4. (R1:4) To ensure a lightweight and computationally efficient design, our model relies solely on raw CAN-based functional data without any image conversion or external sensor inputs such as smartphone acceleration or gravity signals used in DriverBehaviorCNN. As a result, L-CNN demonstrates substantial efficiency gains, achieving a total detection time of 92.375 ms compared to 339.656 ms for DriverBehaviorCNN, despite operating with a larger chunk size (20 ms vs. 1 ms). L-CNN also requires significantly less preprocessing time (29.18 ms) and faster inference (43.195 ms, 3,333 parameters), whereas DriverBehaviorCNN incurs 171.05 ms preprocessing and 173.626 ms inference due to its recurrence-plot image generation and multimodal processing pipeline (2,475,397 parameters). Furthermore, L-CNN is far more memory-efficient, requiring only 112.45 KB compared to 10,226.16 KB for DriverBehaviorCNN. These results confirm that L-CNN provides a more practical and deployable solution for real-time use on resource-constrained embedded platforms such as the Raspberry Pi 4.

VI. Discussion

In deep learning, accurate classification depends heavily on well-defined datasets, while data variability primarily arises from differences in driving behaviors and road conditions. IVN data are inherently dynamic and strongly influenced by individual driving habits and environmental factors. To address these challenges, this study evaluated data collected from 16 Kia drivers and 6 Tesla drivers across diverse driving environments, including urban, city, highway, and restricted regions. Moreover, (R2: 3) to mitigate overfitting, we employed two complementary strategies: dataset diversity and model simplicity. The lightweight CNN architecture, consisting of only 3,333 parameters, inherently restricts model complexity and serves as an effective regularizer. As a result, the proposed system achieves an overall accuracy of approximately 98%–99% with high precision, while maintaining low computational cost.

In addition, detection latency is reduced by approximately 250 ms compared with prior studies, and efficient memory utilization further supports deployment in real-time automotive environments. Finally, (R1:5) validation using electric vehicle (EV) data demonstrates the model's applicability to real-time autonomous systems, where it can monitor and cross-verify control commands, and highlights its suitability for collaborative V2X[3] applications.

While the proposed approach demonstrates strong performance, some limitations require further investigation. (R1: 1) In particular, cross-dataset validation remains an important objective; however, the absence of publicly available benchmarks jointly labeled for both aggressive driving and cyber-attacks currently prevents such evaluation. In addition, achieving a more balanced optimization between classification accuracy and detection latency could be further supported with access to proprietary CANdbc files. Such access would enable deeper analysis of additional correlated operational signals, including engine throttle position, engine coolant temperature, and steering wheel movement, as well as their temporal dependencies and sequential generation cycles. (R1: 1) Although cross-driver and cross-scenario payload similarity was observed, noticeable variations persist due to individual driving behaviors and road conditions. Furthermore, the absence of restricted demographic information limits our ability to fully evaluate cross-driver generalization across factors such as age, gender, and nationality; this will be addressed in future work in compliance with automotive safety standards (R1:11, 12) (e.g., ISO 26262, ISO/SAE 21434).

VII. Conclusion

This work demonstrates that CAN-based analysis can serve as a unified foundation for both driving behavior monitoring and in-vehicle cybersecurity. By designing a lightweight L-CNN architecture and relying exclusively on raw CAN features, the proposed framework achieves real-time detection of aggressive driving and intrusion activities with minimal computational overhead. Experimental results across multiple vehicles and scenarios confirm that high detection accuracy can be attained without complex preprocessing or external sensors, validating the practicality of CAN-only intelligence for embedded deployment. More importantly, the framework establishes that safety and security can be co-analyzed within a single operational pipeline, eliminating the need for separate monitoring systems and reducing system-level complexity. As vehicle architectures continue to evolve toward software-defined and electric platforms, the proposed approach offers a scalable foundation for future safety-critical monitoring systems that are efficient, adaptable, and deployable under real-world constraints.

[3]Vehicle-to-Everything (V2X) communication refers to the exchange of information between a vehicle and its surroundings—including other vehicles (V2V), roadside infrastructure (V2I), pedestrians (V2P), and networks (V2N) [43].

References

[1] A. Adavikottu and N. R. Velaga, "Modeling the impact of driving aggression on lane change performance measures: Steering compensatory behavior, lane change execution duration and crash probability," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 103, pp. 526–553, 2024.

[2] H. Singh and A. Kathuria, "Analyzing driver behavior under naturalistic driving conditions: A review," *Accident Analysis & Prevention*, vol. 150, p. 105 908, 2021.

[3] D. Kim, H. Park, T. Kim, W. Kim, and J. Paik, "Real-time driver monitoring system with facial landmark-based eye closure detection and head pose recognition," *Scientific reports*, vol. 13, no. 1, p. 18 264, 2023.

[4] T. K. Chan, C. S. Chin, H. Chen, and X. Zhong, "A comprehensive review of driver behavior analysis utilizing smartphones," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 10, pp. 4444–4475, 2019.

[5] M. A. Makridis and A. Kouvelas, "Adaptive physics-informed trajectory reconstruction exploiting driver behavior and car dynamics," *Scientific reports*, vol. 13, no. 1, p. 1121, 2023.

[6] L. Liu, Z. Wang, and S. Qiu, "Driving behavior tracking and recognition based on multisensors data fusion," *IEEE Sensors Journal*, vol. 20, no. 18, pp. 10 811–10 823, 2020.

[7] U. Fugiglando et al., "Driving behavior analysis through can bus data in an uncontrolled environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 737–748, 2018.

[8] R. Kumar and A. Jain, "Driving behavior analysis and classification by vehicle obd data using machine learning," *The Journal of Supercomputing*, vol. 79, no. 16, pp. 18 800–18 819, 2023.

[9] E. Lattanzi and V. Freschi, "Machine learning techniques to identify unsafe driving behavior by means of in-vehicle sensor data," *Expert Systems with Applications*, vol. 176, p. 114 818, 2021.

[10] L. Xue et al., "{Said}: State-aware defense against injection attacks on in-vehicle network," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1921–1938.

[11] J. Guo, L. Li, J. Wang, and K. Li, "Cyber-physical system-based path tracking control of autonomous vehicles under cyber-attacks," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 5, pp. 6624–6635, 2022.

[12] D. C. Li, M. Y.-C. Lin, and L.-D. Chou, "Macroscopic big data analysis and prediction of driving behavior with an adaptive fuzzy recurrent neural network on the internet of vehicles," *IEEE Access*, vol. 10, pp. 47 881–47 895, 2022.

[13] S. Milani, H. Khayyam, H. Marzbani, W. Melek, N. L. Azad, and R. N. Jazar, "Smart autodriver algorithm for real-time autonomous vehicle trajectory control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 1984–1995, 2020.

[14] V. Bandur, G. Selim, V. Pantelic, and M. Lawford, "Making the case for centralized automotive e/e architectures," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1230–1245, 2021.

[15] T.-N. Hoang, M. R. Islam, K. Yim, and D. Kim, "Canperfl: Improve in-vehicle intrusion detection performance by sharing knowledge," *Applied Sciences*, vol. 13, no. 11, p. 6369, 2023.

[16] Y. Sun, S. Li, M. Iqbal, and F. Taher, "Securing electric vehicles against can bus replay attacks: A message authentication approach," *IEEE Communications Standards Magazine*, vol. 8, no. 4, pp. 88–95, 2024.

[17] M. R. Islam, M. Sahlabadi, K. Kim, Y. Kim, and K. Yim, "Cf-aids: Comprehensive frequency-agnostic intrusion detection system on in-vehicle network," *IEEE Access*, 2023.

[18] H. Zhang, K. Huang, J. Wang, and Z. Liu, "Can-ft: A fuzz testing method for automotive controller area network bus," in *2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI)*, IEEE, 2021, pp. 225–231.

[19] M. Almehdhar et al., "Deep learning in the fast lane: A survey on advanced intrusion detection systems for intelligent vehicle networks," *IEEE Open Journal of Vehicular Technology*, 2024.

[20] A. Gazdag, S. Lestyán, M. Remeli, G. Ács, T. Holczer, and G. Biczók, "Privacy pitfalls of releasing in-vehicle network data," *Vehicular Communications*, vol. 39, p. 100 565, 2023.

[21] K. Mohammed, M. Abdelhafid, K. Kamal, N. Ismail, and A. Ilias, "Intelligent driver monitoring system: An internet of things-based system for tracking and identifying the driving behavior," *Computer Standards & Interfaces*, vol. 84, p. 103 704, 2023.

[22] M. Shahverdy, M. Fathy, R. Berangi, and M. Sabokrou, "Driver behavior detection and classification using deep convolutional neural networks," *Expert Systems with Applications*, vol. 149, p. 113 240, 2020.

[23] P. Khosravinia, T. Perumal, and J. Zarrin, "Enhancing road safety through accurate detection of hazardous driving behaviors with graph convolutional recurrent networks," *IEEE Access*, 2023.

[24] J. Zhang et al., "A deep learning framework for driving behavior identification on in-vehicle can-bus sensor data," *Sensors*, vol. 19, no. 6, p. 1356, 2019.

[25] S. Bouhsissin, N. Sael, and F. Benabbou, "Driver behavior classification: A systematic literature review," *IEEE Access*, vol. 11, pp. 14 128–14 153, 2023.

[26] C. M. Reddy, A. Anbarasi, N. Mohankumar, I. MV, S. Murugan, et al., "Cloud-based road safety for real-time vehicle rash driving alerts with random forest algorithm," in *2024 3rd International Conference for Innovation in Technology (INOCON)*, IEEE, 2024, pp. 1–6.

[27] S. J. Moore, F. Cruciani, C. D. Nugent, S. Zhang, I. Cleland, and S. Sani, "Deep learning for network intrusion: A hierarchical approach to reduce false alarms," *Intelligent Systems with Applications*, vol. 18, p. 200 215, 2023.

[28] H. K. K. Byung Il Kwak JiYoung Woo, *Driving dataset*, https://ocslab.hksecurity.net/Datasets/driving-dataset, Accessed: 2024-08-29, 2017.

[29] Y. Sahana, A. Gotkhindikar, and S. K. Tiwari, "Survey on can-bus packet filtering firewall," in *2022 International Conference on Edge Computing and Applications (ICECAA)*, IEEE, 2022, pp. 472–478.

[30] K. Yusupov, M. R. Islam, I. Muminov, M. Sahlabadi, and K. Yim, "Adaptive rnn hyperparameter tuning for optimized ids across platforms," *IEEE Open Journal of Vehicular Technology*, 2025.

[31] G. Baldini, "Detection of cybersecurity spoofing attacks in vehicular networks with recurrence quantification analysis," *Computer Communications*, vol. 191, pp. 486–499, 2022.

[32] S. Li, Y. Cao, H. J. Hadi, F. Hao, F. B. Hussain, and L. Chen, "Ecf-ids: An enhanced cuckoo filter-based intrusion detection system for in-vehicle network," *IEEE Transactions on Network and Service Management*, vol. 21, no. 4, pp. 3846–3860, 2024.

[33] S. Longari, D. H. N. Valcarcel, M. Zago, M. Carminati, and S. Zanero, "Cannolo: An anomaly detection system based on lstm autoencoders for controller area network," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1913–1924, 2020.

[34] A. Zhang, Z. Sun, Q. Jiang, K. Wang, M. Li, and B. Wang, "Lipar: A lightweight parallel learning model for practical in-vehicle network intrusion detection," *IEEE Transactions on Intelligent Transportation Systems*, 2025.

[35] H. Sun, J. Wang, J. Weng, and W. Tan, "Kg-id: Knowledge graph-based intrusion detection on in-vehicle network," *IEEE Transactions on Intelligent Transportation Systems*, 2025.

[36] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A survey of deep learning and its applications: A new paradigm to machine learning," *Archives of Computational Methods in Engineering*, vol. 27, pp. 1071–1092, 2020.

[37] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[38] P.-S. T. GmbH, *Pcan-usb x6: 6-channel can and can fd interface for high-speed usb 2.0*, Product webpage, Accessed: 2025-11-15, 2025. [Online]. Available: https://www.peak-system.com/PCAN-USB-X6.438.0.html?&L=1.

[39] A. Adavikottu and N. R. Velaga, "Analysis of factors influencing aggressive driver behavior and crash involvement," *Traffic injury prevention*, vol. 22, no. sup1, S21–S26, 2021.

[40] Tesla, Inc., *Model 3 owner's manual – collision avoidance assist*, Accessed: 2025-12-09, 2025.

[41] G. M. Foody, "Challenges in the real world use of classification accuracy metrics: From recall and precision to the matthews correlation coefficient," *Plos one*, vol. 18, no. 10, e0291908, 2023.

[42] A. Jamshidi et al., "Microrna signature for early prediction of knee osteoarthritis structural progression using integrated machine and deep learning approaches," *Osteoarthritis and cartilage*, vol. 33, no. 3, pp. 330–340, 2025.

[43] V. Maglogiannis, D. Naudts, S. Hadiwardoyo, D. Van Den Akker, J. Marquez-Barja, and I. Moerman, "Experimental v2x evaluation for c-v2x and its-g5 technologies in a real-life highway environment," *IEEE transactions on network and service management.-New York, NY*, vol. 19, no. 2, pp. 1521–1538, 2022.

Md Rezanur Islam received a B.Sc. in Electrical and Electronic Engineering from the University of Asia Pacific, Bangladesh, in 2016, and an M.Sc. in Mobility Convergence from Soonchunhyang University, South Korea, in 2023. He is currently pursuing a Ph.D. in Software Convergence at Soonchunhyang University. His research interests include deep learning, agentic and generative AI–based software development, and in-vehicle network safety and security, with a particular focus on driver state recognition.

Mahdi Sahlabadi an IEEE Senior Member, holds a Ph.D. in Industrial Computing from the National University of Malaysia. His academic journey includes research positions at the Japan Advanced Institute of Science and Technology (JAIST), Singapore Management University (SMU), Sharif University of Tehran (SUT), University Kebangsaan Malaysia(UKM), and Soonchunhyang University (SCH), South Korea. His areas of research interest are process mining, software architecture, cybersecurity, and quality assurance.

Munkhdelgerekh Batzorig is currently pursuing his Ph.D. degree in Information Security at Soonchunhyang University, South Korea. He received the B.S. degree in Information Security from both the Mongolian University of Science and Technology (MUST), Mongolia, and Soonchunhyang University, South Korea, in 2020 through an international dual degree program, and the M.S. degree in Mobility Convergence Security from Soonchunhyang University in 2023. His research focuses on building safe mobility environments through secure cyber-physical communication at the city level, including automotive cybersecurity, intelligent transportation systems, V2X security, and security testing through HILS-based safe testbed environments

Kangbin Yim is a Professor in the Department of Information Security Engineering at Soonchunhyang University, where he has been since 2003. He received his B.S., M.S., and Ph.D. degrees in Electronics Engineering from Ajou University, South Korea, in 1992, 1994, and 2001, respectively. For over 20 years, his primary research has focused on vulnerability identification, threat analysis, and proof-of-concept (PoC) development for both software and hardware. He is also passionate about designing and implementing hardware and software frameworks for system evaluations and commercial services. His recent work has centered on HILS-based dynamic analysis for distributed embedded software, leading a research team of over 30 members in his lab, LISA. Currently, the lab's top priorities include deep-learning-driven analysis of heterogeneous field data with a particular focus on automotive vehicles, industrial control systems, and mobile baseband.