IEEE Open Journal of Vehicular Technology

Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version 27 June, 2024.

Digital Object Identifier 10.1109/OJVT.2024.0627000

Adaptive RNN Hyperparameter Tuning for Optimized IDS Across Platforms

Kamronbek Yusupov*, Md Rezanur Islam*, Ibrokhim Muminov (Member, IEEE)†, Mahdi Sahlabadi (Senior Member, IEEE)‡, and Kangbin Yim‡

[1]* Department of Software Convergence, Soonchunhyang University, Asan-si, Korea
[2]† Department of Computer Software Engineering, Soonchunhyang University, Asan-si, Korea
[3]‡ Department of Information Security Engineering, Soonchunhyang University, Asan-si, Korea

CORRESPONDING AUTHOR: Kangbin Yim (e-mail: yim@sch.ac.kr).

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.

ABSTRACT Modern vehicles are increasingly vulnerable to cyber-attacks due to the lack of encryption and authentication in the Controller Area Network, which coordinates communication between Electronic Control Units. This study investigates the use of Recurrent Neural Networks to improve the accuracy and efficiency of Intrusion Detection Systems in vehicular networks. Focusing on sequential CAN data, we compare the performance of different RNN architectures, including SimpleRNN, LSTM, and GRU, in detecting common attack types like Denial-of-Service, Fuzzing, Replay, and Malfunction. Sixty-three RNN models were tested with various hyperparameters, including optimizers and learning rates. Our findings indicate that GRU models achieve superior detection performance, particularly in resource-constrained environments, offering near 99% accuracy in identifying cyber threats. The study also explores the implications of six different hardware choices, revealing that devices like Jetson and Raspberry Pi, when paired with optimal hyperparameters, can deliver efficient real-time IDS performance at a lower cost. These results contribute to the ongoing effort to secure vehicular communication systems and highlight the importance of balancing accuracy, resource usage, and system cost in IDS deployment.

INDEX TERMS CAN, IVN, Vehicular IDS, RNN Family, Jetson & Raspberry Devices.

I. INTRODUCTION

CONTEMPORARY vehicles are equipped with more than a hundred Electronic Control Units (ECUs) [1] that coordinate the operation of crucial systems such as the engine, brakes, and multimedia control interfaces. ECUs successfully represent the combination of intelligence and comfort while driving, facilitated by the Controller Area Network (CAN) [2], a communication protocol that allows smooth communication between the ECUs. Although the CAN system is widely used for ECU communication, it does not provide encryption and authentication capabilities [3], making vehicles vulnerable to cyber-attacks. Thus, Intrusion Detection Systems (IDSs) have been invented by automotive security specialists as a viable remedy to these vulnerabilities [4], [5]. Through the integration of IDS with the CAN, the security of vehicular communication networks was substantially enhanced [6].

IDS are crucial for ensuring network security since they continuously monitor system activity and network traffic to identify possible cyber threats [7]. For detecting intrusions, IDS use two main techniques: Rule-Based Detection [8] and Non-Rule-Based Detection [9]. Rule-Based Detection in identifying cyber threats utilizes pre-established rules or signatures of known risks [10]. Alerts are triggered when the monitored traffic matches established attack patterns, which are identified based on a dataset of known attack patterns. While this technique is successful in detecting known risk signatures, it lacks the ability to detect novel, unrecognized risks. On the other hand, Non-Rule-Based one detects uncommon patterns by establishing a standard set of regular actions and detecting any deviations from it. Nevertheless, it demonstrates a greater frequency of false positives, despite its capacity to detect novel and unfamiliar attacks.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

Non-Rule-Based Detection is mainly highlighted in this research, which encompasses three types of IDS. The first type is Flow Traffic Analysis IDS. It conducts an analysis of CAN network traffic flow patterns in order to detect anomalies, such as abrupt communications or surges of In-Vehicle Network (IVN) [11]. While this kind of IDS is good at detecting broad patterns of malicious activity [12], cyber-attacks that do not substantially change traffic patterns might remain undetected. On the other hand, Payload IDS uses deep packet inspection to find cyber threats by examining the content of data packets [13]. This technique helps the system identify complex attacks that are hidden inside packets. However, the operation of this device requires significant amount of time and computational power. Hybrid IDS, which combines rule-based and non-rule-based techniques, is generally considered as a solution for securing vehicles from cyber intrusion. It analyzes the correlation between each CAN data, enhancing precision of security breach identification, reducing the occurrence of false positives, and enabling the adjustment of risk to emerging threats [14], [15].

Deep learning techniques are increasingly being used to improve the detection capabilities of Hybrid IDS because of their ability to process large amounts of data and handle complex features. To further enhance the efficiency of deep learning models, it is necessary to use advanced techniques for extracting the most relevant features and feeding the CAN data [16]. As a reason, comprehensive feature extraction and preprocessing guarantee the accuracy and suitability of data in IDS. Moreover, the accurate detection of threats depends on the effectiveness of deep learning models and the optimization of their hyperparameters.

There are two main approaches to provide data to a deep learning model for IDS: the direct input technique [17] and the time-series input method [18]. The direct input technique involves the immediate input of raw data into the models without any processing. However, the effectiveness of IDS might be unintentionally limited by overload problems in deep learning systems functioning in complex environments [19]. The time-series approach, which arranges data in a sequential fashion according to time, offers advantages in handling vast volumes of data and enhancing system performance in situations with substantial data volume, such as IVNs.

Recurrent Neural Networks (RNNs) are particularly suitable for IDS because of the sequential nature of CAN data. This characteristic allows RNNs to effectively handle and learn from this sort of data. Hence, the use of the time-series technique with RNNs results in enhanced performance in IDS [20]–[22].

In order to create an effective IDS for analyzing sequential data, especially for the CAN bus, it is important to consider the following aspect. Consequently, finding the most appropriate RNN models and configurations that can efficiently process sequential data is the main goal of this research. It includes the following tasks: enhancing techniques for extracting features, proposing alternative configurations for separate RNN models, and assessing inference times across different platforms. In addition, this research aims to determine the optimal RNN model that provides high accuracy while minimizing the use of computer resources, by conducting a thorough comparison of many models and their respective outputs on five IoT platforms, such as Jetson AGX Xavier, Jetson Xavier NX, Jetson TX2, Jetson Nano, Raspberry Pi4, and one PC.

Remaining of this paper is organized as follows. Section II examines previous studies on automotive networking and points out the drawbacks of current IDS methodologies. Section III explores the utilization of RNN for feature extraction, providing details on the algorithm, feature extraction process, and data preprocessing techniques. Section IV provides an overview of the outcomes obtained from deploying different RNN models, offering guidance on architecture and hyperparameters, and evaluating model performance in various scenarios. Section V delves into the strengths of our proposed models with a focus on hyperparameters. Lastly, Section VI concludes the paper, summarizing the research findings and discussing the implications.

II. BACKGROUND

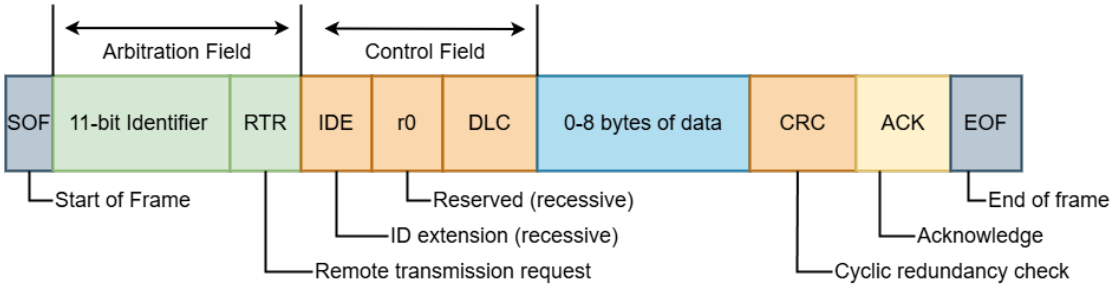
In this section, we discuss the details of the CAN and its variations based on their technical capabilities, highlighting a comparative perspective of CAN 2.0A, CAN 2.0B, and CAN FD. Additionally, we provide an in-depth discussion on Recurrent Neural Networks (RNNs) and their variations, including SimpleRNN, LSTM, and GRU, with a focus on their technical features and a comparative analysis of their performance and applications.

A. Controller Area Network

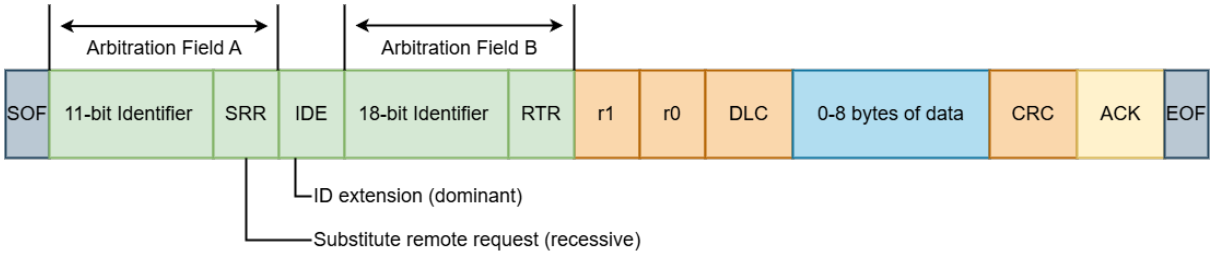
The CAN is a data transmission protocol developed by Bosch in 1983 [23], originally designed to optimize interactions between ECUs in vehicles. The main objectives of CAN were to minimize wiring, improve communication reliability, and simplify the integration of various control systems. CAN represented a significant milestone in the evolution of automotive networks, enabling substantial advancements in both functionality and safety. The first mass production integration of CAN in automotive systems occurred in 1991, after which it became an industry standard.

The CAN is a robust vehicle bus standard designed to enable efficient communication among ECUs. CAN has evolved through three main versions: CAN 2.0A, CAN 2.0B, and CAN FD, each catering to progressively advanced requirements, as detailed in Table 1. CAN 2.0A, as illustrated in Figure 1a, supports a standard identifier [24], limiting unique message IDs and making it suitable for simple control systems with basic communication needs. CAN 2.0B, depicted in Figure 1b, extends the identifier length [25], significantly increasing the number of unique message IDs and enhancing its utility for complex systems requiring

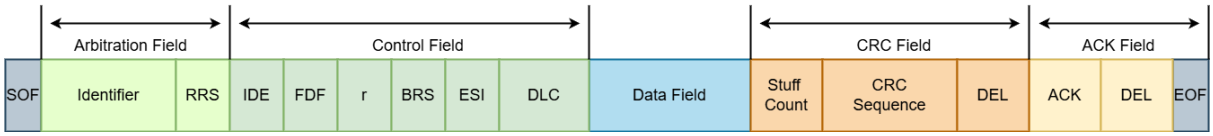
This article has been accepted for publication in IEEE Open Journal of Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJVT.2025.3547761 IEEE Open Journal of Vehicular Technology



(a) **CAN 2.0A: Standard 11-bit Identifier**



(b) **CAN 2.0B: Extended 29-bit Identifier**



(c) **CAN FD: Flexible Data Rate**

FIGURE 1. Comparison of CAN Protocol Versions: CAN 2.0A, CAN 2.0B, and CAN FD

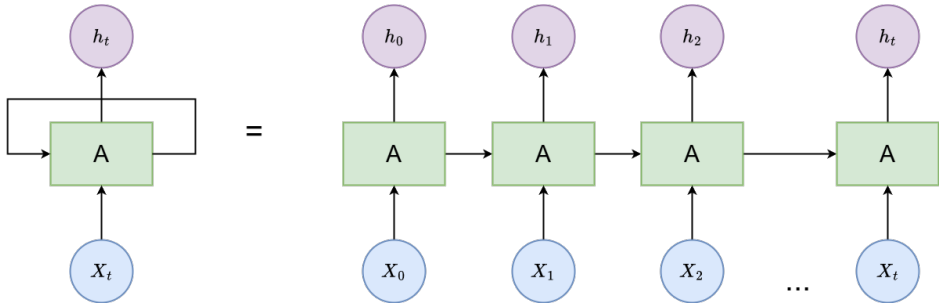
advanced addressing and compatibility. The modern CAN FD (Flexible Data Rate), shown in Figure 1c, introduces higher data transmission speeds and larger payload capacities [26], making it more suitable for the rapid transmission of sensor data required by Advanced Driver Assistance Systems (ADAS) components such as cameras, radars, and LiDARs. These versions are compared in Table 1, which highlights key technical differences, including frame format, data length, transmission speed, and error detection mechanisms. Operating on a CSMA/CR (Carrier Sense Multiple Access with Collision Resolution) mechanism [27], the CAN protocol ensures efficient communication by transmitting messages with identifiers that prioritize critical data, enabling real-time systems to function without delays. This evolution underscores the adaptability of CAN to meet the diverse and growing demands of modern automotive and industrial systems.

CAN is divided into specialized channels to cater to specific vehicle functionalities, such as B-CAN (Body CAN), C-CAN (Chassis CAN), and M-CAN (Multimedia CAN) [28], [29]. B-CAN operates at lower data rates and prioritizes non-critical communication for systems like climate control, lighting, and central locking, where latency is less critical. C-CAN is designed for chassis-related functions that require higher data rates and lower latency, such as brake control, suspension, and power steering. These systems are critical for vehicle safety and performance, making C-CAN essential in modern vehicle designs. M-CAN, on the other hand, handles multimedia and infotainment systems, managing features like audio, video, and navigation data. M-CAN typically operates at intermediate data rates to balance bandwidth needs and system performance. These CAN channels are structured to optimize communication within the IVN by assigning speed and priority levels based on the criticality of the systems they manage, ensuring efficiency and reliability in various vehicle operations.

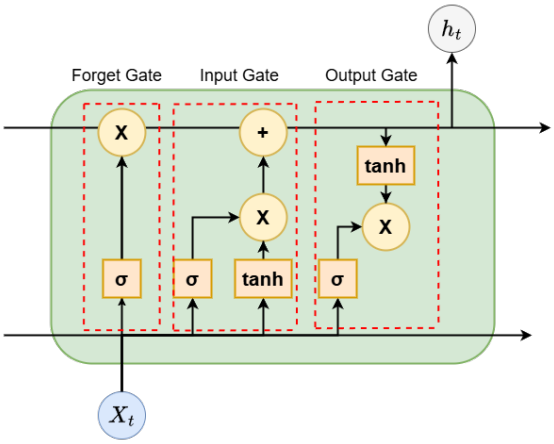
As displayed in Fig 1c, the Start of Frame (SOF) marks the beginning of a new transmission with a single dominant bit, synchronizing all nodes on the CAN bus. The Identifier (ID) field varies between CAN versions, providing message priority through an 11-bit identifier in CAN 2.0A, a 29-bit extended format in CAN 2.0B, and backward-compatible formats in CAN FD. The Remote Transmission Request (RTR) and Identifier Extension (IDE) fields define the type of frame and distinguish between standard and extended identifiers, with CAN FD replacing RTR with the Frame Data Field (FDF) for flexible data rates. Reserved bits like r0, r1, and others, fixed as dominant in CAN 2.0A and

TABLE 1. Technical Differences Between CAN 2.0A, CAN 2.0B, and CAN FD

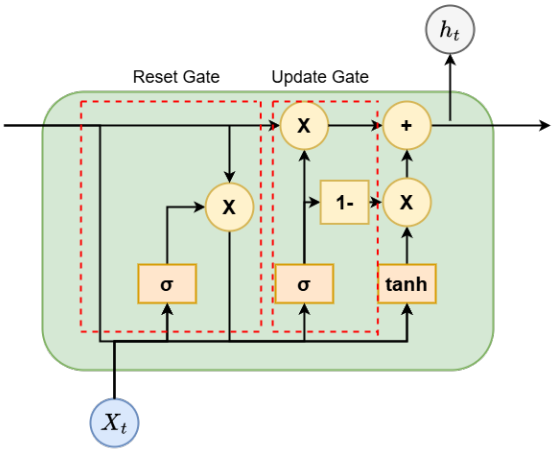
Feature	CAN 2.0A	CAN 2.0B	CAN FD
Frame Format	Standard 11-bit Identifier	Extended 29-bit Identifier	Standard and Extended (Backward Compatible)
Data Length	Up to 8 bytes	Up to 8 bytes	Up to 64 bytes
Data Transmission Speed	Up to 1 Mbps	Up to 1 Mbps	Up to 8 Mbps (arbitration phase retains CAN speed)
Protocol Efficiency	Fixed data rate, high overhead	Fixed data rate, high overhead	Flexible data rate, reduced overhead
Error Detection	15-bit CRC	15-bit CRC	17-bit or 21-bit CRC (enhanced robustness)
Arbitration Method	Non-destructive bitwise arbitration	Non-destructive bitwise arbitration	Non-destructive bitwise arbitration
Backward Compatibility	N/A	Yes (to CAN 2.0A)	Yes (to CAN 2.0)
Bus Load Handling	Limited	Limited	Enhanced (due to increased payload capacity)
Use Case Examples	Basic ECUs and simple sensors	Advanced ECUs, extended addressing	ADAS and high-speed, high-reliability systems
Security Features	None	None	Optional security extensions



(a) **RNN: Recurrent Neural Network Architecture**



(b) **LSTM: Long Short-Term Memory Architecture**



(c) **GRU: Gated Recurrent Unit Architecture**

FIGURE 2. Comparison of RNN Family

2.0B, evolve in CAN FD to enable higher speeds and backward compatibility through additions like FDF and Bit Rate Switching (BRS) [30]. Other essential components include the Data Length Code (DLC), which specifies payload sizes ranging from 0 to 8 bytes in CAN 2.0A and 2.0B, and up to 64 bytes in CAN FD for high-bandwidth needs. The Payload (Data Field) contains the transmitted data, while the Cyclic Redundancy Check (CRC) ensures integrity, offering a 15-bit CRC in CAN 2.0A and 2.0B, and a more robust 17- or 21-bit CRC in CAN FD. The Acknowledge (ACK) field confirms successful transmissions, with dominant or recessive bits indicating acknowledgment status. The frame concludes with the End of Frame (EOF), a 7-bit recessive field, and the Intermission (IFS), a 3-bit spacing that separates consecutive frames and ensures processing readiness.

In designing an effective Intrusion Detection System (IDS) for CAN networks, it is essential to account for the diverse features present in CAN messages, which can vary across different protocol versions such as CAN 2.0A, CAN 2.0B, and CAN FD. These variations influence the input features

that can be utilized for intrusion detection. In this study, we focus on identifying the optimal combination of features that are effective across all versions of CAN messages. This approach aims to develop a standardized IDS framework that ensures robust and reliable detection capabilities, regardless of the underlying CAN protocol version or application. Such a unified system is critical for enhancing the security and resilience of in-vehicle networks across diverse automotive and industrial implementations.

B. Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNN) are a class of artificial neural networks developed in the 1980s for processing time-dependent and sequential data. Unlike traditional feed forward neural networks, which assume independence between input data, RNNs leverage a mechanism of recursion to incorporate information from previous time steps [31]. This makes them ideal for analyzing sequential data such as text, time-series data, and audio signals. However, as mentioned in Table 2, the basic RNN architecture has limitations related to training on long sequences [32], which has led to the development of improved versions like Long Short-Term Memory (LSTM) [33] and Gated Recurrent Unit (GRU) [34]. SimpleRNN shown in Fig. 2a, is the basic architecture of recurrent neural networks. It includes a hidden state that is updated at each time step based on the current input and the state from the previous step. The update formula for the hidden state (h_t) is as follows:

$$h_t = \sigma(W_{xh} \cdot x_t + W_{hh} \cdot h_{t-1} + b_h),$$

where W_{xh} and W_{hh} are weight matrices, b_h is a bias vector, and σ is an activation function, such as $\tanh$.

SimpleRNN performs well on tasks that require capturing short-term dependencies. However, it faces significant challenges when training on long sequences due to vanishing and exploding gradients during Backpropagation Through Time (BPTT) [35]. This limitation restricts the model's ability to effectively learn long-term dependencies, making it less suitable for complex sequential tasks. To address the issue of vanishing gradients, the LSTM architecture was introduced. Unlike SimpleRNN, LSTM incorporates specialized components called "gates" that manage the flow of information, as shown in Fig. 2b. These gates allow the model to preserve important long-term dependencies while filtering out irrelevant data. The main components of LSTM include "forget gate", "input gate", and "output gate". The input gate decides which new information should be stored from the current input. On the other hand, the forget gate determines which information from the previous state should be discarded. Output gate is responsible for regulating which part of the cell state contributes to the current output.

The update of the cell state (c_t) and the hidden state (h_t) in LSTM is defined by the following equations:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \quad i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i),$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c),$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \quad h_t = o_t \cdot \tanh(c_t).$$

where f_t, i_t, and o_t represent the forget gate, input gate, and output gate, respectively, and σ denotes the sigmoid activation function, which constrains values between 0 and 1.

The forget gate

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

decides the proportion of the previous cell state (c_{t-1}) to retain by evaluating the previous hidden state (h_{t-1}) and the current input (x_t).

The input gate

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

determines which parts of the new information, modulated by a tanh activation

$$\tanh(W_c \cdot [h_{t-1}, x_t] + b_c),$$

are relevant to add to the cell state.

The cell state (c_t) is updated by combining the retained information from the forget gate

$$f_t \cdot c_{t-1}$$

and the new information from the input gate

$$i_t \cdot \tanh(...).$$

The output gate

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

controls how much of the updated cell state contributes to the hidden state (h_t), which is further processed through a tanh activation function

$$h_t = o_t \cdot \tanh(c_t).$$

These mechanisms enable LSTM to handle long-term dependencies effectively, making it particularly useful in tasks such as text analysis and prediction of time series. However, the complexity of the LSTM model increases computational costs, which can be a critical factor in resource-constrained environments.

GRU was proposed as a simplified version of LSTM shown in Fig. 2c. GRU reduces computational complexity by combining the forget and input gates into a single "update gate" (z_t) and introducing a "reset gate" (r_t) to control the flow of information. The update equations in GRU are as follows:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z), \quad r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r),$$
$$\tilde{h}_t = \tanh(W_h \cdot [r_t \cdot h_{t-1}, x_t] + b_h),$$
$$h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \tilde{h}_t.$$

Where z_t and r_t represent the update gate and reset gate, respectively, and σ is the sigmoid activation function, which

TABLE 2. Comparison of SimpleRNN, LSTM, and GRU

Feature	SimpleRNN	LSTM	GRU
Architecture	Basic RNN with a single hidden state updated at each time step.	Incorporates forget, input, and output gates for controlling information flow.	Combines forget and input gates into an update gate and adds a reset gate for simpler structure.
Memory Capability	Limited to short-term dependencies due to vanishing gradients.	Handles long-term dependencies effectively with controlled memory gates.	Handles long-term dependencies similarly to LSTM but slightly reduced for very long sequences.
Gradient Issues	Suffers from vanishing and exploding gradients in long sequences.	Mitigates vanishing gradients through gate mechanisms.	Also mitigates vanishing gradients but with a simpler design.
Computational Cost	Low computational complexity due to simplicity.	High computational cost due to complex gates.	Lower computational cost than LSTM, suitable for resource-constrained environments.
Flexibility	Less flexible, suitable for tasks requiring short-term memory.	Highly flexible, suitable for complex and long-term sequential tasks.	Flexible and optimized for environments with limited computational power.
Performance	Limited performance on long sequences.	High performance for tasks requiring long-term memory.	Comparable to LSTM with fewer parameters and faster training.
Applications	Suitable for simple tasks like basic time-series prediction.	Ideal for complex tasks such as language modeling, translation, and sequence forecasting.	Effective for resource-constrained tasks, real-time applications, and mobile systems.

outputs values between 0 and 1 to control the degree of information flow.

The update gate

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

determines how much of the previous hidden state (h_{t-1}) should be retained and carried forward.

The reset gate

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

controls how much of the previous hidden state should be ignored when computing the candidate hidden state ($\tilde{h}_t$).

The candidate hidden state

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \cdot h_{t-1}, x_t] + b_h)$$

is computed by combining the reset-modulated previous hidden state and the current input (x_t), processed through a tanh activation function.

Finally, the hidden state

$$h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \tilde{h}_t$$

is updated as a weighted combination of the previous hidden state ($z_t \cdot h_{t-1}$) and the candidate hidden state ($(1 - z_t) \cdot \tilde{h}_t$), with the weights controlled by the update gate.

This simplified structure of GRU effectively manages long-term dependencies while reducing computational complexity compared to LSTM.

Due to its simplified structure, GRU requires fewer resources while maintaining a performance comparable to that of LSTM. GRU is particularly effective in tasks with limited computational resources and has shown excellent results in sequential data processing.

In this study, we utilize all three types of RNNs, SimpleRNN, LSTM, and GRU, to identify the most effective IDS solution for in-vehicle networks, leveraging their unique strengths and addressing their respective limitations to ensure optimal performance and reliability.

III. RELATED WORK

A systematic review of related studies [41] shows that deep learning-based approaches, especially Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) from the RNN family, were determined to be the most efficient in detecting attacks on vehicles. The reason is that LSTM and GRU effectively handle sequential data tasks and learn long-term dependencies with high accuracy, minimizing the risk of vanishing gradients. Due to its unique architecture, LSTM can control the flow of information through internal gates, helping retain relevant information and discard unnecessary data. GRU models are a simplified version of LSTMs that also effectively handle long-term data while using fewer computational resources, making them suitable for the vehicle IDS.

Previous studies that focused on enhancing IDS through RNN-family models and provide comprehensive details about the research technique, as shown in Table 3, were gathered from the existing literature. To be more specific, the selection criteria for the existing literature are based on feature extraction techniques, RNN model types, input data, attack types, and, especially, details about the inference time on various devices where the tests were conducted. This resulted in the identification of five studies closely related to this study: three using LSTMs and two using GRUs.

Zhu *et al*. proposed a multi-task intrusion detection method [36] for IVN using LSTM and Mobile Edge Computing (MEC). The proposed model includes two separate LSTM neural networks processing data based on temporal and content features. The multi-task LSTM architecture includes a shared hidden layer processed locally, after which data and temporal features are processed in parallel on the MEC server. The effectiveness of this model was confirmed with a high accuracy of 89.3%. Furthermore, the research indicated that the average run time used for each CAN detection took only 0.61 ms. Despite the high detection speed, the use of MEC might cause network delays when transmitting CAN messages to the server. Another research [37] developed CAN-ADF (The Controller Area Network - Attack Detection Framework) to address multi-class clas-

TABLE 3. Summary of Previous Studies on Inference Time and Accuracy in Vehicular IDS Using RNN Models

Author	Type	Dataset	Feature Extraction	Classification Type	Device	Inference Time	Accuracy%
Zhu *et al*. [36]	LSTM	Self-Created Dataset	Payload	Binary Classification	Intel i5 GPU 3.3GHz	0.61 ms	89.3
Tariq *et al*. [37]	LSTM	Self-Created Dataset	CAN ID and Payload	Multiclass Classification	CPU Intel Xeon E5-1650 3.60GHz, GPU GTX 1080Ti	73 ms	99.54
Khan *et al*. [38]	LSTM	UNSWNB-15, Car Hacking Dataset (HCRL)	PCA on full dataset	Binary Classification	Intel i5 3.20 GHz	0.023 ms	98-99
Kukkala *et al*. [39]	GRU	SynCAN	Payload	Binary Classification	Jetson TX2 CPU ARM Cortex-A57	80 ms	99
Haoyu *et al*. [40]	GRU	HRCL	Payload Sum, Time Variance, Time Gap, MAD	Multiclass Classification	Jetson Xavier NX	890 ms	99

sification tasks aimed at detecting intrusions in the CAN communication system. This model is considered hybrid as it combines rule-based detection methods and LSTM to identify anomalies in vehicular communication systems. The attack detection accuracy of the proposed model reached up to 99% with the attack detection time of 73 ms. Khan *et al*. proposed an advanced multi-stage deep learning system for detecting malicious activities in autonomous vehicles [38]. The proposed approach combines two methods, such as the Bloom filter and bidirectional LSTM, to ensure security and provide highly accurate anomaly detection in the CAN communication system. Several methods, including feature transformation, reduction using principal component analysis (PCA), and standardization, were applied for thorough data preprocessing. It has consequently resulted in the anomaly detection accuracy of 98.88% and 99.11% for the UNSW-NB15 and Car Hacking Dataset, respectively. Although a temporal analysis of the testing showed that anomaly detection took only 0.023 ms, this method is considered highly resource-intensive for an IDS, which may pose challenges in environments with limited computational power.

In terms of the studies that implemented a GRU for enhancing a vehicle's IDS, Kukkala *et al*. presented an IDS based on an autoencoder using the GRU model [39] for automotive sensor networks. In the study, they compared their model with previously proposed LSTM-based autoencoders. The results demonstrated that their model outperforms LSTM in detection speed while maintaining the same high accuracy. The results also showed that the developed IDS can detect anomalies in the CAN network with 99% accuracy. However, it is worth noting that the use of autoencoders and GRU requires significant computational resources. The analysis and anomaly detection were performed using a Jetson TX2 device, providing a detection time of 80 ms. Thus, despite the high efficiency and speed of the proposed solution, its implementation requires powerful hardware to ensure stable operation in real-world conditions. On the other hand, Haoyu Ma *et al*. developed a lightweight IDS that utilizes a GRU neural network [40]. Proposed GRU model demonstrated high accuracy, achieving F1-scores of 0.9992 for DoS, 0.9922 for Fuzzy, and 0.9963 for Spoofing attacks. IDS was tested for intrusion detection time on the Jetson Xavier NX device using 5000 CAN messages. The model required 890 ms to analyze and detect all these messages, which points out the challenge of ensuring real-time performance on embedded devices with limited computational power, indicating the need for further simplification or improved hardware capabilities. In conclusion, this study takes a broader approach to the aspects outlined in Table 1. Whereas most of the aforementioned studies utilized self-created CAN data, a standard HRCL dataset was chosen for this research. As a reason, the reliability of synthetic data is a point of discussion in terms of data representativeness, privacy preservation, and, most importantly, potential biases [42]. Furthermore, minimal elements of the CAN data, Time Gap and CAN ID, were chosen for feature extraction, as they can cover all CAN Protocol Attack types, as shown in Table 4. Consequently, multiclass classification had to be selected for each proposed RNN model. This study primarily tested both LSTM and GRU models with various hyperparameter configurations across different IoT devices, identifying the optimal model configuration for each device.

IV. EXPERIMENTAL SETUP

A. FEATURE EXTRACTION

The datasets used in this study are sourced from HCRL's [43] KIA SOUL vehicles and has the following features. They include four different types of attacks mentioned in Table 4 along with an attack-free dataset. These datasets underwent rigorous analysis, cleaning, and standardization. Key features include "Time offset" and "CAN ID". CAN ID defines the payload's priority along with the sender and receiver ECUs' information. Time offset represents the payload generation time via CAN ID. To streamline model training and reduce computational complexity, the study focused on extracting features solely from the CAN ID and the time interval between messages. The dataset was then transformed using a LabelEncoder. This preprocessing step was critical for enhancing the performance of RNN models in the subsequent analysis.

B. PLATFORM SELECTION

The platforms listed in Table 5 were utilized for the practical implementation of an intrusion detection system on in-vehicle networks where all devices are commonly used as single broad computer as shown in related works. These

TABLE 4. Table of CAN Protocol Attack Types and Characteristics

Type of Attack	Description	Method	Example CAN IDs	Timing Constraints	Effects
DoS	Interferes with regular operation by flooding the network with high-priority messages.	Floods the CAN network causing network congestion and eventual failure.	`0x00`	0.0003 to 0.0005 seconds	Disruption of normal system functioning, potential dangerous driving conditions.
Fuzzing	Identifies system weaknesses by sending false information including various priority messages.	Sends identification messages with priorities ranging from high to low.	`0x000` to `0x7FF`	0.0003 to 0.0005 seconds	Reveals system weaknesses, potential malfunction.
Replay	Steals and retransmits valid communications to cause confusion.	Retransmits messages seen as original, causing unintended actions or system setting changes.	N/A	0.0003 to 0.0005 seconds	Confusion inside the system, potential unintended actions.
Malfunction	Sends false signals to manipulate data field values.	Randomly selects CAN IDs and alters data fields.	`0x153`	17 false signals every 5-20 seconds	Anomalous system responses, potential system failure.

devices, ranging from low-power single-board computers like the Raspberry Pi 4, Jetson Nano, and Jetson NX, to more advanced platforms such as the Jetson TX2, Jetson AGX Xavier, and a high-performance PC, provide a diverse range of computational capabilities. The specifications include varying CPU architectures, GPU processing power, and memory bandwidths, enabling the evaluation of IDS performance across different hardware configurations. This setup ensures a comprehensive analysis of the IDS under resource-constrained environments, such as those in embedded automotive systems, as well as high-performance systems for real-time detection.

C. DATA PRE-PROCESSING

In this research, mean normalization was utilized to enhance the performance of RNN within the context of a CAN bus IDS. The preparation of data is a critical step that directly impacts the quality of input data, subsequently improving model training efficacy and boosting the accuracy of predictions. By applying mean normalization technique, the data is centered around zero and scaled according to the standard deviation, which helps in compensating for variations in feature scales and reduces the risk of numerical instability during model training. Standardizing the dataset using these calculations ensures that the features contribute uniformly to the model, thereby enhancing the learning process.

Further, we applied the Time Series Input technique to effectively manage sequential CAN bus data, optimizing the training process by using ten-line windows. This method enables the model to efficiently identify patterns of malicious and attack free behavior by sequentially analyzing the data. If a cyber-attack is detected in any of the ten consecutive time intervals, all intervals are marked as compromised; otherwise, they are labeled safe. This cyclic process across the dataset allows the model to proficiently classify data segments and recognize attack patterns, making time series analysis essential for detecting CAN bus vulnerabilities.

D. HYPER-PARAMETER CONFIGURATION

In this study, hyperparameter selection was performed using a grid search strategy, following the methodology outlined by Md. Delowar et al. Grid search systematically evaluates combinations of hyperparameters to determine the optimal configuration for a model. This approach tested various learning rates (e.g., 0.0001, 0.001, 0.01, and 0.5) and optimizers, including Adam, AdamW, Nadam, Adadelta, Adagrad, and RMSprop. As noted in Md. Delowar et al.'s findings, lower learning rates such as 0.0001 and 0.001 typically lead to stable training and improved model performance, while higher rates like 0.5 can introduce instability and hinder convergence. Based on this grid search process, the selected configurations for the RNN models were finalized and are detailed in Table 6. These configurations served as the foundation for the experimental evaluation, ensuring a systematic and optimized approach to training the models for intrusion detection in in-vehicle networks. This meticulous hyperparameter tuning process was crucial to achieving robust and reliable performance across diverse system environments.

V. EVALUATION

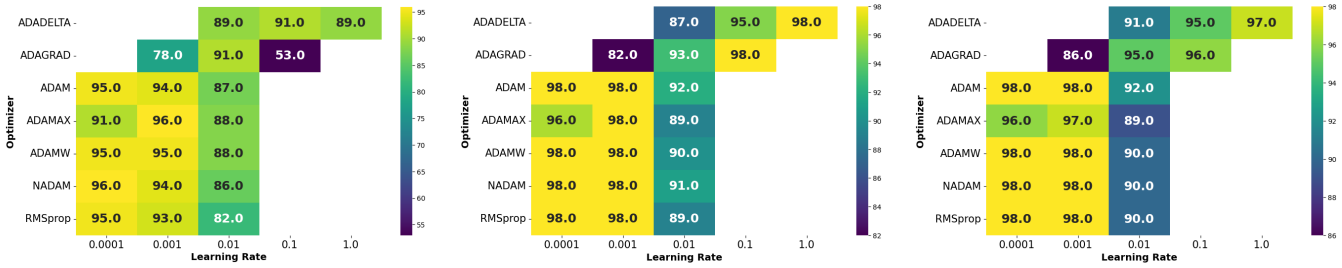
Developing an IDS for IVN with optimal inference time and state-of-the-art accuracy was achieved through a two-stage experiment. In the first stage, sixty-three models using different RNN architectures and hyperparameter settings were evaluated, as shown in Table 6 for hyperparameters and Table 7 for accuracy. The results indicate that for DoS, fuzz, and malfunction attacks, the majority of models attained nearly 100% accuracy. Nevertheless, the accuracy significantly dropped for replay attacks. Therefore, we prioritized each model's ability to identify replay attacks when choosing the optimal option for the IDS, as it was crucial for comprehensive attack detection.

A. ACCURACY MATRIX

In Fig. 3. and Table 8 represent the accuracy metrics of multiple RNN models, on Fig. 3 the x-axis denoting the learning rates and the y-axis representing different optimizers. The SimpleRNN model has lower performance compared to the GRU and LSTM models, as illustrated in Figures 3a, 3c, and 3b, respectively. Significantly, except for settings utilizing a learning rate of 0.001 and optimizers like Adamax, Adagrad, and Adadelta, most models attained an overall accuracy of 98%. 21 of the 63 assessed models shown in Table 7

TABLE 5. Enhanced Device Specifications

Device	CPU Specifications	GPU Specifications	Memory and Bandwidth
Raspberry Pi 4	Quad-core ARM Cortex-A72 Clocked at 1.5 GHz or 1.8 GHz	VideoCore VI Clocked at 600 MHz	8 GB LPDDR4
Jetson Nano	Quad-core ARM Cortex-A57	Maxwell Architecture 128 NVIDIA CUDA cores	4 GB 64-bit LPDDR4
Jetson NX	6-core NVIDIA Carmel ARM®v8.2 64-bit CPU	384-core NVIDIA Volta™ GPU with 48 Tensor Cores	8 GB 128-bit LPDDR4x
Jetson TX2	Dual-core NVIDIA Denver 2 Quad-core ARM Cortex-A57 64-bit CPU	256-core NVIDIA Pascal™ GPU Optimized for power efficiency	8 GB 128-bit LPDDR4
Jetson AGX Xavier	8-core NVIDIA Carmel ARM®v8.2 64-bit CPU 8MB L2 + 4MB L3	512-core NVIDIA Volta™ GPU with 64 Tensor Cores AI acceleration support	32 GB 256-bit LPDDR4x Bandwidth: 136.5 GB/s
PC	AMD Ryzen 5 3600XT 6-Core Clocked at 3.79 GHz	GTX 1660 Super Turing Architecture	16 GB DDR4



(a) **SimpleRNN** (b) **LSTM** (c) **GRU**

FIGURE 3. Performance analysis of RNN models for replay attack detection: a comparative evaluation of optimizers and learning rates

were selected as the ideal RNN-based IDS configuration for further evaluation. The IDS performance evaluation in Table 8 shows that both GRU and LSTM models achieve high accuracy (97-98%) across all configurations, with GRU Nadam slightly outperforming others at 98% accuracy. Precision, recall, and F1-scores for critical attack classes like DoS, Fuzz, and Malfunction consistently reach 1.00, indicating excellent detection capabilities. Replay attacks demonstrate slight variations in performance, with GRU Nadam achieving the highest F1-score of 0.94.

B. INFERENCE TIME

In the second stage of the study, the selected LSTM and GRU models were thoroughly tested on five different IoT platforms and one PC to evaluate their inference time, as shown in Fig. 4. As a result, the Jetson AGX Xavier stands out as the highest performer when paired with the Adam optimizer at a learning rate of 0.001 for both LSTM and GRU, achieving the fastest inference times. However, its higher cost restricts its feasibility for cost-sensitive applications. On the other hand, the Jetson Xavier NX delivers nearly comparable performance with all hyperparameters at a significantly lower price point, making it a more cost-effective alternative. Though Jetson TX2 provide poor performance compared to Jetson Xavier NX. This highlights the critical trade-off between performance and cost, which must be carefully considered when selecting embedded systems for real-time applications.

Furthermore, the performance evaluation across various hyperparameters underscores the importance of tuning for specific hardware. Proper tuning enables even low-cost devices like the Raspberry Pi 4 to achieve near real-time detection, whereas the Jetson Nano did not perform as well compared to other devices. For example, using the Nadam optimizer with a learning rate of 0.001 for the LSTM, and the AdamW optimizer with the same learning rate for the GRU on the Raspberry Pi 4, delivered results comparable to the best-performing setups, despite the Raspberry Pi's limited computational resources. This suggests that careful optimization can achieve an effective balance between cost and performance in embedded systems, particularly with GRU models. While the Jetson AGX Xavier offers superior performance, the cost-effectiveness of the Raspberry Pi 4, especially when paired with appropriate hyperparameters like Nadam and AdamW at a 0.001 learning rate, cannot be missed for budget-sensitive applications. This study emphasizes the vital role of hyperparameter tuning in maximizing performance while managing costs, particularly in resource-constrained environments.

In Fig. 5, a comparative analysis illustrates the deployment of the proposed model across six devices, with their specifications detailed in Table 5. In contrast, most previous studies evaluated their models primarily on PCs, with a few exceptions, such as Haoyu *et al.*, who tested on the Jetson Xavier NX, and Kukkala *et al.*, who utilized the Jetson TX2. However, this study provides a broader evaluation by

TABLE 6. RNN Hyperparameter Configuration

Hyperparameter	Value
Model Type	SimpleRNN GRU LSTM
Input Size	2
Sequence Length	10
Number of RNN Layers	2
Hidden Size (RNN models)	128 (first layer) 64 (second layer)
Dropout Rate	0.2
Batch Normalization	After each RNN layer
Fully Connected Layers	1
Output Layer	Softmax
Loss Function	CategoricalCrossentropy
Optimizer	Adadelta Adagrad Adam Adamax AdamW Nadam RMSprop
Learning Rate	0.0001 0.001 0.01 0.1 1.0
Number of Classes	5
Number of Epochs	50
Batch Size	128

TABLE 7. Accuracy of Different RNN Models Configurations by Optimizer

Optimizer	Learning Rate	SimpleRNN (%)	GRU (%)	LSTM (%)
Adam	LR 0.0001	0.98	0.99	0.99
	LR 0.001	0.98	0.99	0.99
	LR 0.01	0.89	0.94	0.93
Nadam	LR 0.0001	0.97	0.99	0.99
	LR 0.001	0.98	0.99	0.98
	LR 0.01	0.87	0.91	0.92
Adamax	LR 0.0001	0.96	0.98	0.98
	LR 0.001	0.98	0.99	0.99
	LR 0.01	0.89	0.91	0.90
Adadelta	LR 1.0	0.96	0.98	0.99
	LR 0.1	0.97	0.98	0.98
	LR 0.01	0.96	0.96	0.95
Adagrad	LR 0.001	0.91	0.94	0.93
	LR 0.01	0.96	0.98	0.97
	LR 0.1	0.80	0.99	0.99
AdamW	LR 0.0001	0.98	0.99	0.99
	LR 0.001	0.98	0.99	0.99
	LR 0.01	0.90	0.91	0.91
RMSprop	LR 0.0001	0.98	0.99	0.99
	LR 0.001	0.97	0.99	0.99
	LR 0.01	0.89	0.90	0.90

including a wider range of commonly used IoT platform. It is important to note that the performance of models evaluated on PCs cannot be standardized due to the varying specifications of these machines. Haoyu *et al.*'s study on the Jetson Xavier NX model employed a learning rate of 0.01 with the Adam optimizer, achieving an inference time of 890 ms for 5000 sequential inputs. When the model was adjusted to a window size of 10, the inference time was 1.78 ms. In contrast, the proposed model utilizing a GRU with the Nadam optimizer and a learning rate of 0.001 achieved an inference time of approximately 7.5 ms on the Jetson Xavier NX. The re-implementation of Haoyu *et al.*'s model revealed certain limitations. Their approach relies on an extensive feature extraction process but omits time series windowing and excludes Replay attack analysis. Consequently, the model demonstrated a significantly higher false positive rate at around 10% on DoS, Fuzz and Spoofing attack in real-world, dynamic scenarios, underscoring the weakness of their proposed model in handling diverse and evolving conditions.

Similarly, Kukkala *et al.*'s study on the Jetson TX2 with a GRU model used a learning rate of 0.0001 with the Adam optimizer, resulting in an inference time of 80 ms with a window size of 20; with a window size of 10, the inference time was 40 ms. In contrast, the proposed GRU model by this study on the same device, with the AdamW optimizer and a learning rate of 0.001, significantly reduced the inference time to just 7.55 ms. Additionally, when comparing the performance on other devices, such as the Jetson AGX Xavier, Jetson Nano, and Raspberry Pi 4, the proposed model delivered inference times under 10 ms in most cases, demonstrating the efficiency and scalability of this approach across different IoT platform. Moreover, by fine-tuning hyperparameters, such as the learning rate and optimizer, the model's performance can be further enhanced, ensuring efficient operation across various hardware platforms.

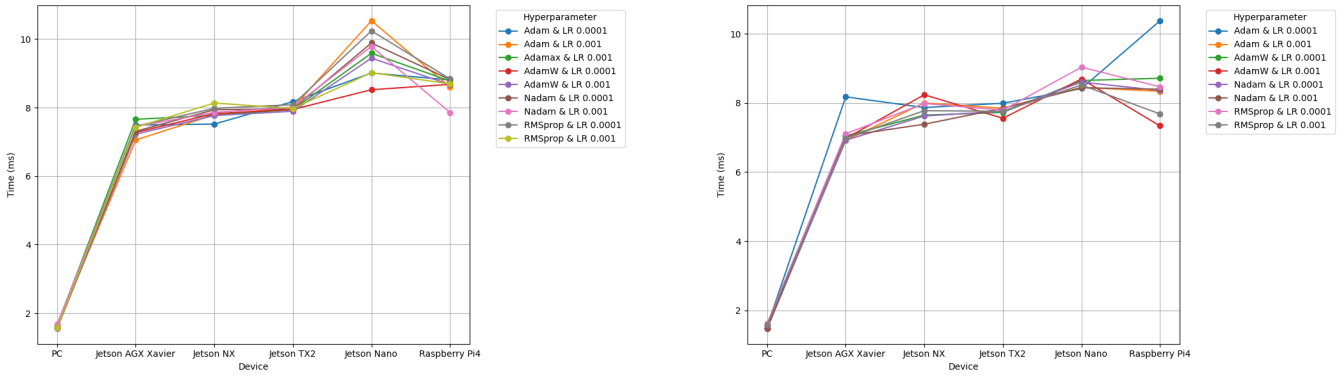
C. CROSS VALIDATION

To evaluate the performance and robustness of the proposed IDS models, a 7-fold Cross-Validation method was applied, as shown in Fig. 6. This approach, chosen for its balance between computational efficiency and performance, divides the dataset containing millions of records into seven equal subsets while maintaining class balance (e.g., normal data and anomalies). In each iteration, six folds are used for training, and the remaining fold is reserved for testing, ensuring that every fold is used as a test set exactly once. This systematic process is repeated seven times, providing a comprehensive assessment of the models' capabilities. The results highlight consistent accuracy across all folds, with GRU achieving a maximum accuracy of 98% using the Nadam optimizer and LSTM performing competitively with up to 97% accuracy. These findings validate the effectiveness of the proposed system in detecting intrusions within in-

This article has been accepted for publication in IEEE Open Journal of Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJVT.2025.3547761 IEEE Open Journal of Vehicular Technology

TABLE 8. Performance Metrics of Models with Different Configurations

Model	Class	Precision	Recall	F1-score	Accuracy	Macro Avg (P/R/F1)	Weighted Avg (P/R/F1)
GRU AdamW	Normal	0.88	0.99	0.93	0.97	0.97	0.97
	DoS	1.00	1.00	1.00			
	Fuzz	1.00	1.00	1.00			
	Replay	0.99	0.87	0.92			
	Malfunction	1.00	1.00	1.00			
LSTM AdamW	Normal	0.93	0.91	0.92	0.97	0.97	0.97
	DoS	1.00	1.00	1.00			
	Fuzz	1.00	1.00	1.00			
	Replay	0.91	0.93	0.92			
	Malfunction	1.00	1.00	1.00			
GRU Nadam	Normal	0.92	0.97	0.94	0.98	0.98	0.98
	DoS	1.00	1.00	1.00			
	Fuzz	1.00	1.00	1.00			
	Replay	0.97	0.91	0.94			
	Malfunction	1.00	1.00	1.00			
LSTM Nadam	Normal	0.94	0.91	0.92	0.97	0.97	0.97
	DoS	1.00	1.00	1.00			
	Fuzz	1.00	1.00	1.00			
	Replay	0.91	0.95	0.93			
	Malfunction	1.00	1.00	1.00			



(a) **LSTM** (b) **GRU**

FIGURE 4. Comparative inference time analysis of RNN models with diverse hyperparameters on five IoT devices and one PC.

vehicle networks, showcasing a reliable and unbiased evaluation methodology.

VI. DISCUSSION

The study demonstrates significant advancements in utilizing RNNs, particularly GRU models, for intrusion detection in IVNs. By optimizing hyperparameters and employing time-series data input based on CAN ID sequence and time interval, which is applicable to all types of CAN versions, the system achieves high accuracy, nearly 99%, for various cyber-attack types such as DoS, fuzz, and malfunction. However, the accuracy for replay attacks is comparatively lower at 94%. Replay attacks are hard to detect because they mimic normal network flow, lacking distinct anomalies, and ML models struggle with their high similarity to normal data. Limited contextual features and under-representation in datasets further reduce detection accuracy. The results showcase that GRU models outperform traditional LSTM models in both inference time and computational efficiency, making them well-suited for resource-constrained environments. Additionally, the study highlights the adaptability of the model across different hardware platforms, offering both cost-effective and high-performance solutions. This capability ensures the model's broader applicability, from low-cost embedded systems to more advanced processing units, all while maintaining high detection rates.

The choice of optimizer and learning rate significantly impacts model performance and accuracy. While optimizers like Adagrad, Adadelta, and RMSprop adapt learning rates over time, they often lack momentum, making them slower and less efficient in resource-constrained environments. Adam is a popular choice due to its adaptive learning rate and momentum, but it can be sensitive to low learning rates, such as 0.0001, which may lead to overfitting and

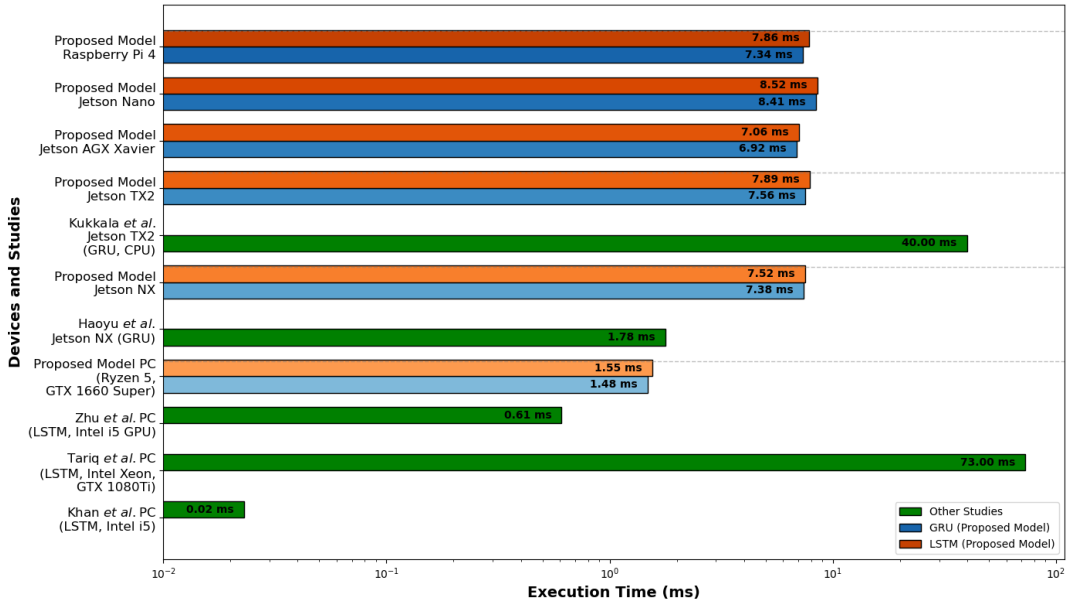


FIGURE 5. Comparative inference time evaluation of vehicular IDS across five IoT devices and one PC, with previous study benchmarks.

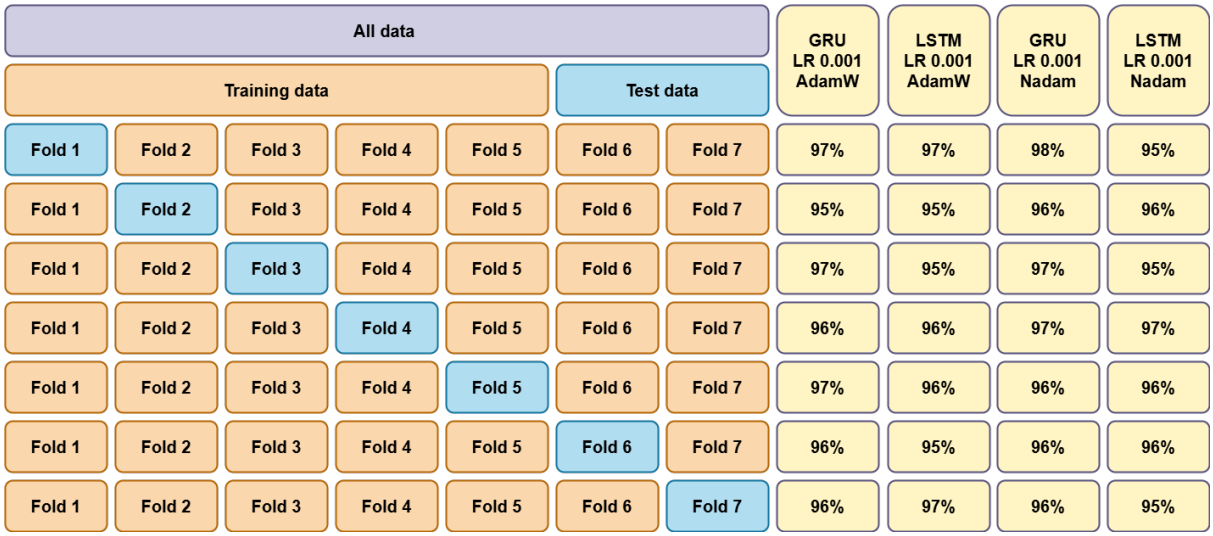


FIGURE 6. IDS performance evaluation using K-Fold cross-validation, demonstrating accuracy results for GRU and LSTM models with different optimizers (AdamW and Nadam) and a learning rate of 0.001 across various folds of training and testing data.

slower convergence. In contrast, Nadam and AdamW, both with a learning rate of 0.001, strike an optimal balance between speed and stability. As a reason, Nadam builds on Adam by incorporating Nesterov momentum, which anticipates the gradient direction before updating, resulting in faster and smoother convergence, particularly on low-resource devices. AdamW, on the other hand, addresses the problem of overfitting by incorporating weight decay, improving generalization without sacrificing momentum. These features make Nadam and AdamW more effective than other optimizers, as they ensure faster updates, better adaptation, and enhanced performance, even in environments with limited computational power. When combined with a learning rate of 0.001, these optimizers consistently deliver superior results in terms of both performance and efficiency.

VII. CONCLUSION

This paper presents an in-depth analysis of deploying IDS using RNNs for IVNs. The study explores the effectiveness of various RNN architectures, such as SimpleRNN, LSTM, and

GRU, in detecting cyber-attacks like DoS, Fuzz, Replay, and Malfunction on the CAN bus. The findings show that GRU models outperform others in terms of both accuracy and computational efficiency, making them particularly suitable for real-time detection in resource-constrained environments. By tuning hyperparameters such as the learning rate and optimizer, the model achieved nearly 99% accuracy across all tested devices, demonstrating its adaptability to different hardware specifications.

One of the limitations of deploying IDS across different IVN-based systems is the variability in hardware specifications. For instance, while high-performance devices like the Jetson AGX Xavier showed excellent results with low inference times, lower-cost devices like the Raspberry Pi 4 required careful hyperparameter tuning to achieve similar results. This variability poses challenges for standardizing IDS deployment across different IVNs. Consequently, future work will focus on developing lightweight IDS solutions tailored to specific IVN architectures and exploring more sophisticated techniques such as federated learning and edge computing to balance performance and resource limitations. Additionally, the integration of emerging security measures for autonomous vehicles, such as encrypted communication protocols, can further enhance the robustness of IDS solutions in automotive environments.

REFERENCES

[1] M. Bozdal, M. Samie, and I. Jennions, "A survey on can bus protocol: Attacks, challenges, and potential solutions," in *2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*. IEEE, 2018, pp. 201–205.

[2] H. M. Song and H. K. Kim, "Discovering can specification using on-board diagnostics," *IEEE Design & Test*, vol. 38, no. 3, pp. 93–103, 2020.

[3] S. Nie, L. Liu, and Y. Du, "Free-fall: Hacking tesla from wireless to can bus," *Briefing, Black Hat USA*, vol. 25, no. 1, p. 16, 2017.

[4] E. Aliwa, O. Rana, C. Perera, and P. Burnap, "Cyberattacks and countermeasures for in-vehicle networks," *ACM computing surveys (CSUR)*, vol. 54, no. 1, pp. 1–37, 2021.

[5] S.-F. Lokman, A. T. Othman, and M.-H. Abu-Bakar, "Intrusion detection system for automotive controller area network (can) bus system: a review," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–17, 2019.

[6] M. Gmiden, M. H. Gmiden, and H. Trabelsi, "Cryptographic and intrusion detection system for automotive can bus: Survey and contributions," in *2019 16th International Multi-Conference on Systems, Signals & Devices (SSD)*. IEEE, 2019, pp. 158–163.

[7] S. Jin, J.-G. Chung, and Y. Xu, "Signature-based intrusion detection system (ids) for in-vehicle can bus network," in *2021 IEEE international symposium on circuits and systems (ISCAS)*. IEEE, 2021, pp. 1–5.

[8] M. H. Shahriar, Y. Xiao, P. Moriano, W. Lou, and Y. T. Hou, "Canshield: Deep-learning-based intrusion detection framework for controller area networks at the signal level," *IEEE Internet of Things Journal*, vol. 10, no. 24, p. 22111–22127, Dec. 2023. [Online]. Available: http://dx.doi.org/10.1109/JIOT.2023.3303271

[9] K. Agrawal, T. Alladi, A. Agrawal, V. Chamola, and A. Benslimane, "Novelads: A novel anomaly detection system for intra-vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 22 596–22 606, 2022.

[10] L. Zhang and D. Ma, "A hybrid approach toward efficient and accurate intrusion detection for in-vehicle networks," *IEEE Access*, vol. 10, pp. 10 852–10 866, 2022.

[11] 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. 21 266–21 289, 2019.

[12] K. Yusupov, M. R. Islam, I. Oh, M. Sahlabadi, and K. Yim, "Security assessment of in-vehicle network intrusion detection in real-life scenarios," in *12th International Conference of Security, Privacy and Trust Management (SPTM 2024)*, ser. Computer Science & Information Technology (CS & IT), D. C. Wyld and D. Nagamalai, Eds., vol. 14, no. 11, Sydney, Australia, June 22–23 2024.

[13] M. Hassan, M. E. Haque, M. E. Tozal, V. Raghavan, and R. Agrawal, "Intrusion detection using payload embeddings," *IEEE Access*, vol. 10, pp. 4015–4030, 2021.

[14] L. Yang, A. Moubayed, and A. Shami, "Mth-ids: A multitiered hybrid intrusion detection system for internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 616–632, 2021.

[15] E. M. Maseno, Z. Wang, and H. Xing, "A systematic review on hybrid intrusion detection system," *Security and Communication Networks*, vol. 2022, no. 1, p. 9663052, 2022.

[16] J. Lee, W. Kim, J.-H. Cho, D. S. Kim, T. J. Moore, F. F. Nelson, and H. Lim, "Deep learning approach for attack detection in controller area networks," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*, vol. 11746. SPIE, 2021, pp. 195–200.

[17] O. Y. Al-Jarrah, K. El Haloui, M. Dianati, and C. Maple, "A novel detection approach of unknown cyber-attacks for intra-vehicle networks using recurrence plots and neural networks," *IEEE Open Journal of Vehicular Technology*, vol. 4, pp. 271–280, 2023.

[18] Y. Kamronbek, I. M. Rezanur, I. Oh, and K. Yim, "Time series mean normalization for enhanced feature extraction in in-vehicle network intrusion detection system," in *International Conference on Broadband and Wireless Computing, Communication and Applications*. Springer, 2023, pp. 302–311.

[19] 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. 100565, 2023.

[20] B. Lampe and W. Meng, "Intrusion detection in the automotive domain: A comprehensive review," *IEEE Communications Surveys & Tutorials*, 2023.

[21] V. Tanksale, "Intrusion detection system for controller area network," *Cybersecurity*, vol. 7, no. 1, p. 4, 2024.

[22] 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.

[23] C. Specification, "Bosch," *Robert Bosch GmbH, Postfach*, vol. 50, p. 15, 1991.

[24] T. C. Dönmez, "Anomaly detection in vehicular can bus using message identifier sequences," *IEEE Access*, vol. 9, pp. 136 243–136 252, 2021.

[25] M. Marchetti and D. Stabili, "Read: Reverse engineering of automotive data frames," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1083–1097, 2018.

[26] Y. Xie, P. Huang, W. Liang, and Y. He, "Comparison between can and can fd: A quantified approach," in *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*. IEEE, 2017, pp. 1399–1403.

[27] M. K. Ishak, C. C. Leong, and E. A. Sirajudin, "Embedded ethernet and controller area network (can) in real time control communication system," in *10th International Conference on Robotics, Vision, Signal Processing and Power Applications: Enabling Research and Innovation Towards Sustainability*. Springer, 2019, pp. 133–139.

[28] 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.

[29] Y. An, J. Park, I. Oh, M. Kim, and K. Yim, "Design and implementation of a novel testbed for automotive security analysis," in *Innovative Mobile and Internet Services in Ubiquitous Computing: Proceedings of the 14th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2020)*. Springer, 2021, pp. 234–243.

[30] N. Lodge, N. Tambe, and F. Saqib, "Addressing vulnerabilities in can-fd: An exploration and security enhancement approach," *IoT*, vol. 5, no. 2, pp. 290–310, 2024.

[31] L. Kozachkov, M. Ennis, and J.-J. Slotine, "Rnns of rnns: Recursive construction of stable assemblies of recurrent neural networks," *Ad-*

vances in neural information processing systems, vol. 35, pp. 30 512–30 527, 2022.
[32] M. Kaur and A. Mohta, "A review of deep learning with recurrent neural network," in *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 2019, pp. 460–465.
[33] F. Laghrissi, S. Douzi, K. Douzi, and B. Hssina, "Intrusion detection systems using long short-term memory (lstm)," *Journal of Big Data*, vol. 8, no. 1, p. 65, 2021.
[34] A. Pudikov and A. Brovko, "Comparison of lstm and gru recurrent neural network architectures," in *International Scientific and Practical Conference in Control Engineering and Decision Making*. Springer, 2020, pp. 114–124.
[35] A. Tjandra, S. Sakti, R. Manurung, M. Adriani, and S. Nakamura, "Gated recurrent neural tensor network," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 448–455.
[36] K. Zhu, Z. Chen, Y. Peng, and L. Zhang, "Mobile edge assisted literal multi-dimensional anomaly detection of in-vehicle network using lstm," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4275–4284, 2019.
[37] S. Tariq, S. Lee, H. K. Kim, and S. S. Woo, "Can-adf: The controller area network attack detection framework," *Computers & Security*, vol. 94, p. 101857, 2020.
[38] I. A. Khan, N. Moustafa, D. Pi, W. Haider, B. Li, and A. Jolfaei, "An enhanced multi-stage deep learning framework for detecting malicious activities from autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25 469–25 478, 2021.
[39] V. K. Kukkala, S. V. Thiruloga, and S. Pasricha, "Indra: Intrusion detection using recurrent autoencoders in automotive embedded systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3698–3710, 2020.
[40] 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*, vol. 2022, no. 1, p. 5827056, 2022.
[41] F. Luo, J. Wang, X. Zhang, Y. Jiang, Z. Li, and C. Luo, "In-vehicle network intrusion detection systems: a systematic survey of deep learning-based approaches," *PeerJ Computer Science*, vol. 9, p. e1648, 2023.
[42] Y. Lu, M. Shen, H. Wang, X. Wang, C. van Rechem, and W. Wei, "Machine learning for synthetic data generation: a review," *arXiv preprint arXiv:2302.04062*, 2023.
[43] OCSLAB, "Ocs: Open cyber security lab datasets," https://ocslab.hksecurity.net/.

Kamronbek Yusupov received his B.S. degree in Information Security Engineering from Soonchunhyang University, South Korea, in 2023. He is currently pursuing his M.S. degree in Software Convergence at Soonchunhyang University, South Korea. His research interests include deep learning, cybersecurity for in-vehicle networks (IVN), anomaly detection, and big data analysis, reflecting his commitment to utilizing advanced technologies in these areas.

Md Rezanur Islam received 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. Currently pursuing a Ph.D. in Software Convergence at Soonchunhyang, his research focuses on deep learning, anomaly detection, malware detection, computer vision, with a specialization in driver state recognition.

Ibrokhim Muminov a member of IEEE Intelligent Transportation Systems Society (ITSS), IEEE Computer Society (CS), and an IEEE Student Member, is currently pursuing a Bachelor of Science degree in Computer Software Engineering as a Korean Government Scholarship recipient at Soonchunhyang University, South Korea. His research interests encompass Big Data Analysis, Artificial Intelligence, and Computer Vision, with a focus on driver state recognition. He is passionate about developing innovative solutions to enhance driver and vehicle safety through advanced technologies.

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.

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.