electronics

Article

Application-Layer Anomaly Detection Leveraging Time-Series Physical Semantics in CAN-FD Vehicle Networks

Rui Zhao [1], Cheng Luo [2], Fei Gao [1,3,*], Zhenhai Gao [1,3], Longyi Li [1], Dong Zhang [4] and Wengang Yang [5]

[1] College of Automotive Engineering, Jilin University, Changchun 130025, China; rzhao@jlu.edu.cn (R.Z.); gaozh@jlu.edu.cn (Z.G.); lilongyi2001@163.com (L.L.)
[2] School of Automotive Studies, Tongji University, Shanghai 201804, China; cluo@tongji.edu.cn
[3] State Key Laboratory of Automotive Simulation and Control, Jilin University, Changchun 130025, China
[4] Department of Mechanical and Aerospace Engineering, Brunel University London, Uxbridge UB8 3PH, UK; dong.zhang@brunel.ac.uk
[5] Ji Hua Laboratory, Foshan 528251, China; yangwg@jihualab.ac.cn
* Correspondence: gaofei123284123@jlu.edu.cn

Abstract: The Controller Area Network with Flexible Data-Rate (CAN-FD) bus is the predominant in-vehicle network protocol, responsible for transmitting crucial application semantic signals. Due to the absence of security measures, CAN-FD is vulnerable to numerous cyber threats, particularly those altering its authentic physical values. This paper introduces Physical Semantics-Enhanced Anomaly Detection (PSEAD) for CAN-FD networks. Our framework effectively extracts and standardizes the genuine physical meaning features present in the message data fields. The implementation involves a Long Short-Term Memory (LSTM) network augmented with a self-attention mechanism, thereby enabling the unsupervised capture of temporal information within high-dimensional data. Consequently, this approach fully exploits contextual information within the physical meaning features. In contrast to the non-physical semantics-aware whole frame combination detection method, our approach is more adept at harnessing the physical significance inherent in each segment of the message. This enhancement results in improved accuracy and interpretability of anomaly detection. Experimental results demonstrate that our method achieves a mere 0.64% misclassification rate for challenging-to-detect replay attacks and zero misclassifications for DoS, fuzzing, and spoofing attacks. The accuracy has been enhanced by over 4% in comparison to existing methods that rely on byte-level data field characterization at the data link layer.

Keywords: anomaly detection; physical semantics; timing prediction; CAN-FD

check for updates

Citation: Zhao, R.; Luo, C.; Gao, F.; Gao, Z.; Li, L.; Zhang, D.; Yang, W. Application-Layer Anomaly Detection Leveraging Time-Series Physical Semantics in CAN-FD Vehicle Networks. *Electronics* **2024**, *13*, 377. https://doi.org/10.3390/electronics13020377

Academic Editor: Myung-Sup Kim

Received: 5 December 2023
Revised: 10 January 2024
Accepted: 15 January 2024
Published: 16 January 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the progression of information technology and the rise of the Internet of Things (IoT), intelligent connected vehicles (ICVs) have emerged as the forefront of automobile evolution. ICVs represent a next-generation breed of vehicles, outfitted with cutting-edge chips and sensors, seamlessly integrated with the Internet, and endowed with sophisticated environmental perception and advanced intelligent control systems. As vehicle interfaces proliferate and the frequency of information exchange between automotive systems and the external environment intensifies, the vulnerability of the automobile network to hacker intrusions significantly escalates. The in-vehicle Controller Area Network with Flexible Data-Rate (CAN-FD) network plays a pivotal role in facilitating real-time data exchange among multiple sensors, actuators, and ECUs situated within critical subsystems. This network is responsible for transmitting crucial physical information, such as vehicle speed, brake pressure, and the operational status of each subsystem. Vehicle manufacturers segment this data into distinct fields within the CAN-FD payload in accordance with application layer requirements. Consequently, a single CAN-FD data frame can encapsulate multiple physical attributes.

Potential security vulnerabilities in ICVs enable attackers to breach the CAN-FD network, gaining access to essential ECUs for the purpose of reading and writing through external interfaces, including physical connections (e.g., OBD II and cellular interfaces), short-range wireless connections (e.g., Bluetooth and Wi-Fi), and long-range wireless connections (e.g., over-the-air updates and GPS), as illustrated in Figure 1. Such compromise could impact specific services within the vehicle, such as the powertrain and body/chassis server. Given that CAN-FD messages encapsulate a substantial amount of information with tangible physical significance, this information is integral for actuator control. Moreover, due to the absence of robust security mechanisms, CAN-FD networks remain susceptible to a range of attacks, including denial-of-service (DoS) attacks [1], fuzzing attacks [2,3], replay attacks [4,5], and spoofing attacks [6]. These attacks involve injecting attack messages to manipulate their authentic physical meaning. In Figure 1, the first 16 bits of the message with ID 0x304 have been tampered with. Its physical meaning is linked to the speed of the vehicle and, in accordance with the communication matrix definition, can be translated into a speed signal value. The last frame with ID 0x304 in the message corresponds to a vehicle speed of 5.76 m/s. However, the tampered message now corresponds to a vehicle speed of 22.76 m/s, introducing a potential risk of loss of speed control and posing a grave threat to vehicle safety. To ensure the stability and safety of vehicle operations, it is imperative to detect anomalies within the CAN-FD network, particularly focusing on any irregularities in the physical characteristics associated with each field in the CAN-FD message. This aspect is fundamental to maintaining the normal functionality of vehicle operations.

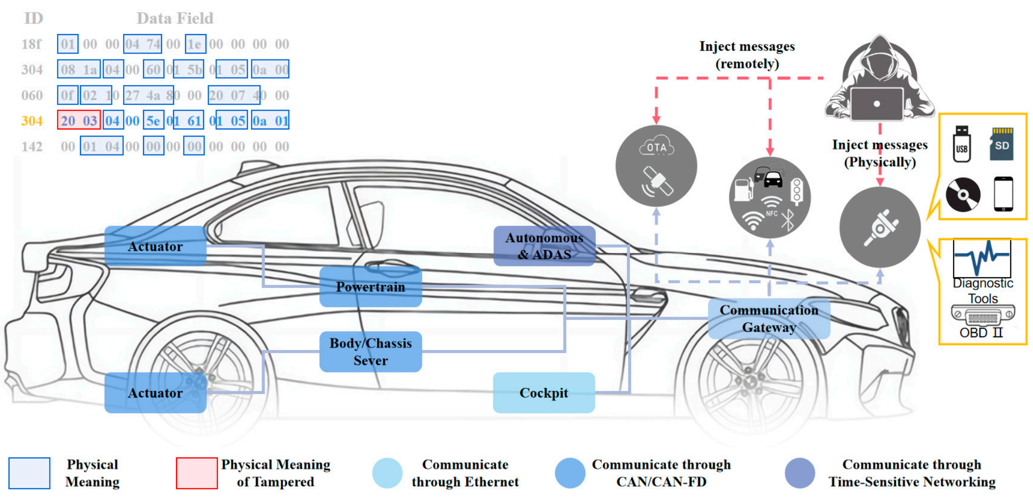

ID
Data Field
18f 01 00 00 04 74 00 1e 00 00 00 00
304 08 1a 04 00 60 01 5b 01 05 0a 00
060 0f 02 10 27 4a 80 00 20 07 40 00
304 20 03 04 00 5e 01 61 01 05 0a 01
142 00 01 04 00 00 00 00 00 00 00 00
Inject messages (remotely)
Inject messages (Physically)
OTA
NFC
USB
SD
Diagnostic Tools
OBD II
Actuator
Autonomous & ADAS
Powertrain
Body/Chassis Sever
Communication Gateway
Actuator
Cockpit
Physical Meaning
Physical Meaning of Tampered
Communicate through Ethernet
Communicate through CAN/CAN-FD
Communicate through Time-Sensitive Networking

Figure 1. In-vehicle network architecture and attack surface: The CAN/CAN-FD bus functions as the primary network application in both the powertrain and body/chassis domains within in-vehicle network architectures, exposing attack surfaces to potential threats, including near-physical contact, short-range wireless, and long-range wireless attacks.

Numerous studies have applied cryptographic algorithms, such as digital signatures, encryption, and message authentication codes, to improve message security in both CAN-FD and its foundational version, CAN. Farag [7] employs synchronized key generators across all nodes to dynamically alter the symmetric key, facilitating the encryption of the 8-byte payload data of a CAN message. This algorithm ensures the uniqueness of the encrypted message at any given time, rendering it effective in preventing replay attacks. Jo et al. [8] propose the MAuth-CAN authentication protocol to counter masquerade attacks on the CAN bus. Xie et al. [9] enhance security by incorporating message authentication codes into CAN-FD messages, achieved by promptly discarding most sequences to establish a lower bound for the on-board application and employing a generous interval from the lower bound to the deadline. Subsequently, they introduced the two-pointer moving rule to dynamically adjust the message authentication code size of each CAN-FD message, ensuring real-time performance [10]. However, the communication overhead associated with these techniques is generally high, posing challenges for their deployment in demanding real-time CAN bus networks.

In recent years, numerous intrusion detection algorithms have been proposed for CAN buses, categorized into periodicity-based and data-domain-based methods based on their detection scope and foundation. Periodicity-based methods leverage the regularity of CAN messages to identify potential anomalous behaviors within the CAN bus. Electronic Control Units (ECUs) typically generate CAN messages at specific frequencies, and deviations in message transmission frequency [11,12] or arrival interval time [13,14] can signal potential anomalies introduced by external attackers injecting messages. Normal, attack-free CAN messages exhibit standard or stable entropy, leading some researchers to propose entropy-based methods for anomaly detection in CAN [15,16]. External attacks on the CAN bus can induce abnormal changes in the physical attributes of the ECU, prompting the utilization of clock drift [17,18], clock skew [19], or voltage [20–22] of the ECU for anomaly detection. Olufowobi et al. [23] detect anomalies in the network based on the real-time schedulable response time of the CAN bus, while Marchetti et al. [24] construct transformation matrices using a standard CAN dataset, comparing sequences of attacked CAN message IDs to detect attacks. Yu et al. [25] establish and validate the network topology through simple random wandering-based network topology to identify external intrusion devices. Despite their effectiveness in determining the normality of message periodic properties, the aforementioned periodicity-based methods cannot identify anomalies within the data domain of the message.

Data-field-based anomaly detection methods identify anomalies by predicting the data fields of the message. Wang et al. [26] introduce an anomaly detection method based on hierarchical temporal memory (HTM) learning, wherein pre-decoded binary data streams are input into individual data sequence predictors. The output predictions are subsequently processed by an anomaly scoring mechanism. Taylor et al. [27] employ an LSTM network on CAN message data fields for prediction, utilizing prediction errors as signals for anomaly detection in the sequence. Dong et al. [28] construct a multi-observation Hidden Markov Model (HMM) based on the ID and data fields of normal CAN bus traffic. This model calculates the probability of frame existence within a defined time window and identifies anomalies based on whether the probability exceeds a threshold value. Zhang et al. [29] develop the CAN message graph, integrating statistical message sequences with message contents to construct and train Graph Neural Networks (GNNs) suitable for directed attribute graphs to predict intrusions. However, the aforementioned data-domain-based anomaly detection methods typically rely on the entire frame or individual bytes within the frame as the detection unit, resulting in a coarser detection granularity or potential segmentation of the complete physical meaning of the application. This may impede the machine's ability to effectively learn and comprehend relevant physical features, posing challenges to improving detection accuracy.

CAN-FD presents clear advantages over CAN, including elevated data transfer rates, expanded data frames, and flexible data rates, positioning CAN-FD as a gradual replacement for conventional CAN. Current anomaly detection endeavors have concentrated on traditional CAN, with minimal consideration directed towards CAN-FD. The increased size of data frames implies that CAN-FD harbors greater physical meaning within its data fields. Given that each distinct physical meaning exhibits a unique pattern of variation, approaching the comprehensive data frame as a singular unit for detection poses a challenge in precisely delineating the normal and abnormal characteristics inherent in this heterogeneous application data. To enhance the precision and granularity of anomaly discrimination in CAN-FD networks, a shift from the data link layer to a higher network application layer is essential. Fine-grained detection for each application's physical field within the CAN-FD data frame becomes necessary to accurately identify anomalies in the genuine physical meaning of each key feature. As a result, a Physical Semantics-Enhanced Anomaly Detection (PSEAD) method for CAN-FD data fields is proposed.

The contributions of this paper can be summarized as follows:

1. Utilizing distinct physical meaning extraction rules for various ID messages, this approach extracts the genuine physical meaning features present in the data fields of CAN-FD messages. This enables the model to effectively identify abnormal variations in the vehicle's key physical parameters. Additionally, based on the number of data types, the features associated with different CAN-FD IDs are reorganized and consolidated. The rearranged features undergo one-hot encoding and minimum-maximum scaling, significantly mitigating the dimensionality expansion introduced by one-hot encoding while preserving the numerical relationships intrinsic to the physical meanings.
2. A novel CAN-FD timing prediction model, Multiple self-attention mechanisms for time-series prediction, which combines the multi-head self-attention mechanism with an LSTM network, is introduced. This model is well-suited for forecasting the physical meaning of CAN-FD messages, offering lower computational overhead and excellent real-time performance. By incorporating the multi-head self-attention mechanism, the model adeptly directs its attention to various segments within the input sequence, facilitating the acquisition of temporal feature representations across distinct hierarchical levels and granularities. This architectural choice empowers the model to holistically capture sequence information, thereby enhancing the precision of abnormal traffic detection within CAN-FD networks.
3. Differing from data sources generated in simulated environments, our approach for the first time employs a dataset collected from an actual vehicle, utilizing a genuine communication protocol matrix to parse the physical meanings in CAN-FD messages. Meanwhile, attack datasets were curated by considering the mechanisms and distinctive characteristics of various attack methods. The experimental outcomes demonstrate that our method attains an accuracy improvement exceeding 4% when compared with the non-physical content-aware whole-frame combination detection method.

The remainder of this paper is structured as follows: Section 2 provides an overview of the CAN-FD message data frame composition, in-vehicle network architecture, various types of attacks, and problem definitions; Section 3 outlines the data preprocessing methods and delineates the construction of the attack dataset; Section 4 delves into the intricate design particulars of the constructed multiple self-attention mechanisms for the time-series prediction model; Section 5 focuses on the performance evaluation of the anomaly detection model; and, finally, the paper draws its conclusions in Section 6.

2. Background Information and Problem Definition

2.1. CAN-FD Background

As automotive electronic systems have grown in complexity, the traditional CAN bus protocol has faced limitations in terms of bandwidth and transmission rate. In response, CAN-FD was developed as an extension of CAN. CAN-FD offers support for data phase bit rates of up to 10 Mbps, accompanied by 64-byte payloads for messages. This extension of data fields [30] empowers CAN-FD with several advantages over CAN, including faster object pool transfer, reduced bus load usage, shorter worst-case response times, and lower jitter [31]. These characteristics render CAN-FD particularly well-suited for application scenarios where substantial data transfer is required, such as Advanced Driver Assistance Systems (ADAS) and autonomous driving. The composition of a complete CAN-FD standard data frame, as shown in Figure 2, comprises several key components:

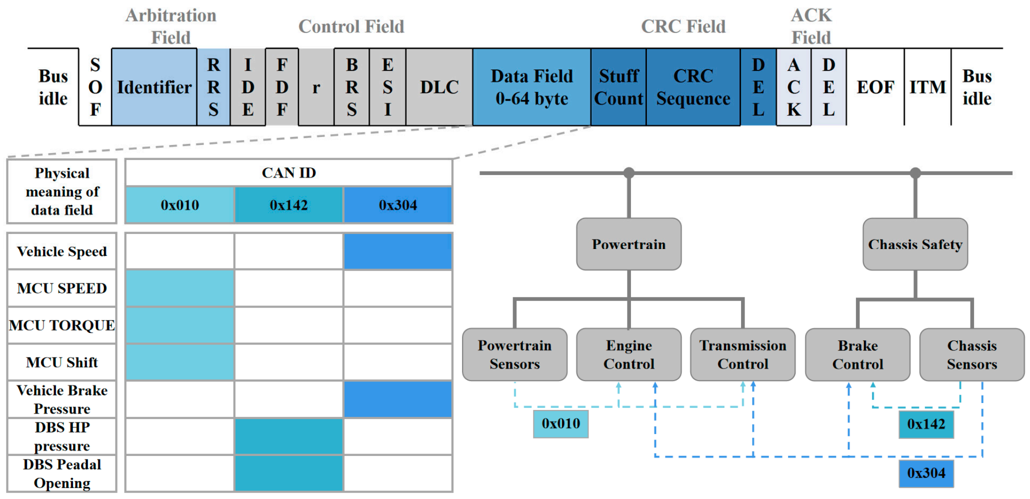


Figure 2. CAN-FD frame structure and physical meaning of the data field.

- **Start of Frame (SOF)**: This signals the beginning of data frame transmission.
- **Arbitration Field**: This field consists of the ID identifier and the RRS bit, which serve as the identification number of the receiving ECU and indicate the message's priority. Generally, a lower ID value indicates a higher message priority.
- **Control Field**: This section provides control over various aspects of the data frame.
- **Data Field**: The core part of the message, it carries the actual data and is interpreted by the receiving ECU to discern its true physical meaning. In the case of CAN-FD, this field can hold up to 64 bytes of data.
- **CRC Check Field**: This field aids in detecting transmission errors. The sending node computes the CRC value and places it in this field, while the receiving node calculates the CRC using the same algorithm and compares it with the received CRC value to determine the accuracy of the message transmission.
- **ACK Answer Field**: All receiving nodes provide an ACK response to confirm whether the data frame was transmitted correctly.
- **End of Frame (EOF)**: This bit marks the conclusion of the data frame.

Within the in-vehicle network, communication protocols such as CAN-FD primarily operate at the physical layer and data link layer. Automakers typically need to construct application layer protocols atop the data link layer. These three layers collaborate to facilitate the exchange of data and information among the internal components of the vehicle system connected to the CAN-FD network. The application layer protocol is responsible for defining and overseeing the data to be transmitted. For instance, it establishes the ID and the rules for the division and filling of the data domain fields. Additionally, it conveys information regarding the true physical significance of the vehicle's speed, engine rpm, and brake pressure. To enhance cost-effectiveness, resource utilization, and network maintenance, it is often necessary to aggregate multiple signals interacting with the same ECU into a single CAN-FD frame, as illustrated in Figure 2. For instance, a message with ID 0x010 encompasses data such as shift, motor feedback speed, and torque. Both the sender and receiver of a CAN-FD message interpret the data based on a universal communication matrix. The sender of the message with ID 0x010 is the ECU responsible for powertrain, while the receivers include the ECUs associated with engine control and transmission control. The physical and data link layer communication capabilities provided by CAN-FD are then utilized to transmit this data.

2.2. Type of Attack

The broadcast nature of the CAN-FD network allows all ECUs connected to the bus to access the transmitted messages in the network. Upon receiving a CAN-FD message, each ECU arbitrates whether to accept the message based on its ID. However, the ID alone cannot authenticate the origin of a CAN-FD message without supplementary security measures. Attackers can exploit this security vulnerability to inject abnormal traffic into the CAN-FD network through various wired and wireless connections, posing a grave threat to vehicle security. Common attacks on CAN-FD networks encompass the following:

- **DoS Attack**: Crafted with the intent to impede authorized entities from accessing resources or to introduce delays in the operation of time-critical systems [32]. This disrupts the ability of other ECUs in the vehicle to process legitimate messages properly, potentially leading to interference, blockage, or paralysis of the CAN-FD bus. Consequently, the vehicle may become unable to function normally or carry out essential tasks.
- **Fuzzing Attack:** Involves the injection of randomly or semi-randomly generated data into a target system, aimed at scrutinizing the system's response under the influence of anomalous or invalid inputs [33]. This is achieved by sending a large number of abnormal or random CAN-FD messages, which can result in erratic vehicle behavior. Symptoms may include actions such as steering wheel shaking, irregular activation of signal lights, or unexpected gear shifting [2].
- **Replay Attack**: Involves the recording of data from a node over a specific time interval, followed by its subsequent replay at a later point in time [34]. Legitimate CAN-FD messages are intercepted and subsequently resent to the CAN-FD bus to execute malicious operations or disrupt the vehicle's normal operation. This can lead to the repetition of actions or inappropriate operations by the vehicle system, potentially endangering vehicle safety.
- **Spoofing Attack**: By transmitting counterfeit data frames to the targeted node with the aim of impersonating legitimate data or commands from an authentic source [35]. By simulating legitimate ECU communication through forged CAN-FD messages, attackers can spoof other ECUs or the vehicle control system. This can result in a loss of control over the vehicle, unintended operations, or confusion within the vehicle's systems.

2.3. Problem Definition

The anomaly detection challenge in the CAN-FD network can be addressed by employing a network model that has acquired knowledge of typical timing variations to forecast messages. If the variance between the observed message and the predicted outcome is substantial, it can be inferred that an anomaly exists within the observed message. Assuming the training dataset is denoted as $\boldsymbol{M}_s$, the feature extraction process can be represented as follows:

$$\boldsymbol{X}_s = \Phi(\boldsymbol{M}_s) \tag{1}$$

where $\boldsymbol{X}_s$ represents the features extracted from the training set and Φ denotes the feature extraction function. The process of training the network model, denoted as Ψ, can be expressed as follows:

$$\overset{\sim}{\Psi} = \Gamma(\Psi, \boldsymbol{X}_s) \tag{2}$$

where Ψ represents the initial network model, $\overset{\sim}{\Psi}$ stands for the trained network model, and Γ symbolizes the training process. Assuming that a specific count of CAN-FD messages before the moment t is represented as $\boldsymbol{l}_t$, where the feature $\boldsymbol{X}_{in,t} = \Phi(\boldsymbol{l}_t)$, and a specific count of CAN-FD messages after the moment t is denoted as $\boldsymbol{l}_{t+1}$, with the feature $\boldsymbol{X}_{out,t} = \Phi(\boldsymbol{l}_{t+1})$. The prediction of the message change utilizing $\overset{\sim}{\Psi}$ can be expressed as follows:

$$\overset{\sim}{\boldsymbol{X}}_{out,t} = \overset{\sim}{\Psi}(\boldsymbol{X}_{in,t}) \tag{3}$$

where $\widetilde{X}_{out,t}$ represents the anticipated feature of the CAN-FD message at a subsequent time interval, the loss value is computed by comparing $\widetilde{X}_{out,t}$ with $X_{out,t}$:

$$\theta_t = Loss\left(\widetilde{X}_{out,t}, X_{out,t}\right) \tag{4}$$

The threshold is defined as θ; if $\theta_t < \theta$, M_{t+1} is considered normal, otherwise, an anomaly is detected.

3. Data Preprocessing

3.1. Data Acquisition

The initial attack-free CAN-FD dataset $M_{Benign} = \left(m_1, m_2, \ldots, m_{N_{Benign}}\right)^T$ was gathered, comprising all CAN-FD messages transmitted on the vehicle's CAN bus during a specific time interval. This dataset includes information such as timestamps of CAN-FD messages, IDs, data fields, and their lengths. The distribution of individual ID messages is illustrated in Figure 3a, while the variation in different CAN ID data fields is depicted in Figure 3b. To facilitate training and testing, part of the originally collected CAN-FD dataset is partitioned into the training set $M_s = (m_{s,1}, m_{s,2}, \ldots, m_{s,N_s})^T$, while the remaining data is designated as the test set $M_t = (m_{t,1}, m_{t,2}, \ldots, m_{t,N_t})^T$. Here, m signifies a frame of message instances, and m_s and m_t represent message instances in the training and test sets, respectively.

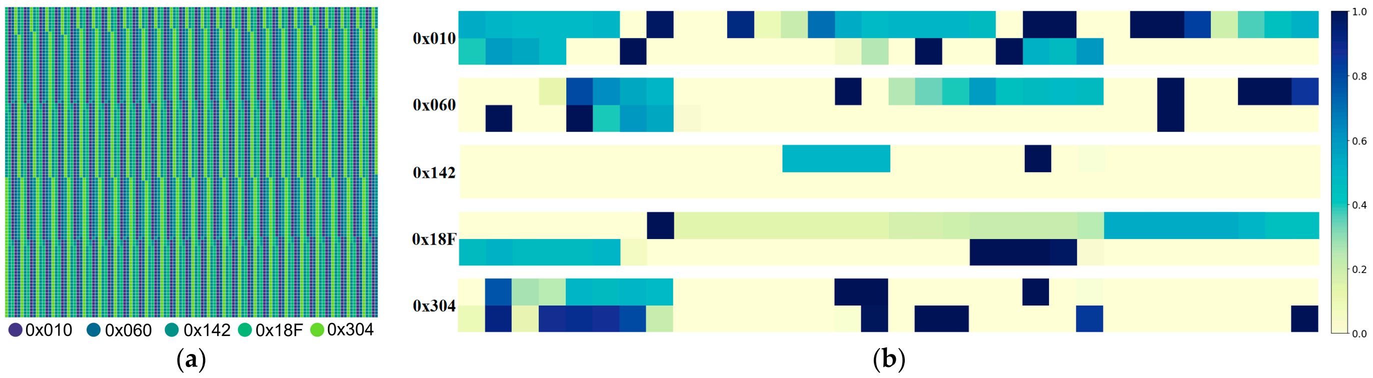


Figure 3. Examples of the distribution of messages with varying IDs in the training set and changes in data fields: (**a**) various colors depict the distribution of messages with distinct IDs within the training set; (**b**) various colors represent alterations in the corresponding bits within the data field. Light yellow signifies that the bit remains consistently at 0, dark blue indicates that the bit remains consistently at 1, while other colors denote bits that are subject to change.

3.2. Data Augmentation

Data augmentation is a technique that expands the volume of trainable data by introducing modified versions of existing data or generating new data based on the existing dataset [36]. In our work, data augmentation was applied to the training set by altering the sliding step size of the original time-series data window and changing the sequence order of the original time-series data. Data augmentation effectively amplifies the diversity of data, intending to empower the model to grasp the inherent data domain invariants more adeptly [37]. This augmentation not only boosts the model's generalization capabilities but also fortifies its robustness, thereby mitigating the risk of overfitting. The size of the sliding window for the input message sequence of the training set, denoted as L_s^{in}, is represented by u, and the sliding step is denoted as z. Additionally, the size of the output message sequence L_s^{out}, is represented as v. In this context, the following relationships apply:

$$L_s^{in} = \left(l_{s,1}^{in}, l_{s,2}^{in}, \ldots, l_{s,p}^{in}\right)^T \tag{5}$$

$$\boldsymbol{L}_s^{out} = \left(\boldsymbol{l}_{s,1}^{out}, \boldsymbol{l}_{s,2}^{out}, \ldots, \boldsymbol{l}_{s,p}^{out}\right)^T \tag{6}$$

$$p = ceil\left(\frac{N_s - u - v}{z}\right) \tag{7}$$

where $\boldsymbol{l}_{s,i}^{in}$ and $\boldsymbol{l}_{s,i}^{out}$ ($i \in [1,2, \ldots, p]$) represent pairs of $\boldsymbol{l}_{s,i}^{in}$ and $\boldsymbol{l}_{s,i}^{out}$ corresponding to each other's input and output sequences, respectively. The "ceil" function denotes the ceiling function, which rounds a number up to the nearest integer. These relationships satisfy the following conditions:

$$\boldsymbol{l}_{s,i}^{in} = (m_{s,1+iz}, m_{s,2+iz}, \ldots, m_{s,u+iz})^T \tag{8}$$

$$\boldsymbol{l}_{s,i}^{out} = (m_{s,1+u+iz}, m_{s,2+u+iz}, \ldots, m_{s,v+u+iz})^T \tag{9}$$

The sliding interval of the window can be controlled by employing various step sizes z_i (where $i \in [1,2, \ldots, N_{step}]$, and N_{step} represents the total number of steps). In this manner, a broader range of input sequences $\boldsymbol{L}_{s,z_i}^{in}$ and corresponding target output sequences $\boldsymbol{L}_{s,z_i}^{out}$ can be generated based on the original data. This variation in step sizes allows for the creation of multiple input-output sequence pairs, providing a more diverse set of training data.

$$\boldsymbol{L}_{s,z_i}^{in} = \begin{bmatrix} m_{s,1+z_i} & m_{s,2+z_i} & \ldots & m_{s,u+z_i} \\ m_{s,1+2z_i} & m_{s,2+2z_i} & & m_{s,u+2z_i} \\ & \vdots & \ddots & \vdots \\ m_{s,1+lz_i} & m_{s,2+lz_i} & \cdots & m_{s,u+lz_i} \end{bmatrix} \tag{10}$$

$$\boldsymbol{L}_{s,z_i}^{out} = \begin{bmatrix} m_{s,1+u+z_i} & m_{s,2+u+z_i} & \ldots & m_{s,v+u+z_i} \\ m_{s,1+u+2z_i} & m_{s,2+u+2z_i} & & m_{s,v+u+2z_i} \\ & \vdots & \ddots & \vdots \\ m_{s,1+u+lz_i} & m_{s,2+u+lz_i} & \cdots & m_{s,v+u+lz_i} \end{bmatrix} \tag{11}$$

The training set data augmentation method is implemented in Algorithm 1, with the required initialization parameters including the training set $\boldsymbol{M}_s$, various sliding step sizes z, the number of shuffle disruptions $N_{shuffle}$, the size of the sliding window u, and the size of the output sequence v. The algorithm proceeds as follows: It begins by generating the input sequence $\boldsymbol{L}_{s,z_i}^{in}$ and the corresponding output sequence $\boldsymbol{L}_{s,z_i}^{out}$ for different sliding step sizes z_i. The input and output sequences created with different sliding steps are then randomly disrupted in correspondence, with this process being repeated $N_{shuffle}$ times. This operation combines the input and output sequences produced in the previous step to obtain the data-enhanced input sequence $\boldsymbol{L}_{s,augment}^{in}$ and output sequence $\boldsymbol{L}_{s,augment}^{out}$. For data augmentation of the training set, we employed a sliding window step z = [1–3]. Subsequently, we sequentially merged the datasets generated by these three distinct sliding window steps. Following this, we subjected the merged dataset to two random disruptions. Finally, we merged the undisturbed dataset with the dataset generated by the two random disruptions, resulting in the ultimate data-augmented dataset. The aforementioned additions have been annotated in blue within the latest manuscript.

Algorithm 1: Data augmentation

```
Input: Training dataset $\boldsymbol{M}_s$, different slide step lengths $z$, number of disruptions $N_{shuffle}$, sliding window size $u$, output sequence size $v$
Output: Input sequence $\boldsymbol{L}^{in}_{s,augment}$ and output sequence $\boldsymbol{L}^{out}_{s,augment}$ after data augmentation.
$\boldsymbol{L}^{in}_{s,augment}$ = empty,$\boldsymbol{L}^{out}_{s,augment}$ = empty
For $z_i$ in $z$ do:
    $\left(\boldsymbol{L}^{in}_{s,z_i}, \boldsymbol{L}^{out}_{s,z_i}\right) \leftarrow$calculate the input sequence and output sequence $(\boldsymbol{M}_s, u, v, z_i)$
    append $\boldsymbol{L}^{in}_{s,z_i} \rightarrow \boldsymbol{L}^{in}_{s,augment}$,$\boldsymbol{L}^{out}_{s,z_i} \rightarrow \boldsymbol{L}^{out}_{s,augment}$
End
For $k$ in range $\left(N_{shuffle}\right)$ do:
        $\left(\boldsymbol{L}^{in}_{s,shuffle_k}, \boldsymbol{L}^{out}_{s,shuffle_k}\right) \leftarrow$disrupt in a random and corresponding order
        $\left(\boldsymbol{L}^{in}_{s,augment}, \boldsymbol{L}^{out}_{s,augment}\right)$
End
For $k$ in range $\left(N_{shuffle}\right)$ do:
        append $\boldsymbol{L}^{in}_{s,shuffle_k} \rightarrow \boldsymbol{L}^{in}_{s,augment}$,$\boldsymbol{L}^{out}_{s,shuffle_k} \rightarrow \boldsymbol{L}^{out}_{s,augment}$
End
Return $\left(\boldsymbol{L}^{in}_{s,augment}, \boldsymbol{L}^{out}_{s,augment}\right)$
```

3.3. Construction of the Attack Dataset

Based on the attack characteristics, this section outlines algorithms for generating attack datasets to simulate scenarios where an attacker conducts DoS, fuzzing, replay, and spoofing attacks against the CAN-FD network, as depicted in Algorithm 2. The initialization parameters needed for creating the attack dataset include the test set $\boldsymbol{M}_t$, the total number of selected data segments N_{attack}, the number of messages contained in each data segment n^{attack}, the range of data frames to be inserted $\left[k^{attack}_{min}, k^{attack}_{max}\right]$, and the attack messages m^{attack}. To determine the insertion locations of the attack messages, the algorithm starts by randomly choosing non-overlapping data segments n^{attack} from the test set $\boldsymbol{M}_t$. These selected non-overlapping data segments are denoted as $\boldsymbol{M}^{attack,i}_t(i \in [1, 2, \ldots, N_{attack}])$. Subsequently, the attack dataset $\boldsymbol{M}^{attack}$ is formed by randomly inserting k^{attack} frames of attack messages m^{attack}, where k^{attack} is a random number within the range $\left[k^{attack}_{min}, k^{attack}_{max}\right]$, after each message frame within these data segments. The parameters N_{attack}, n^{attack}, k^{attack}_{min}, k^{attack}_{max}, and m^{attack} can be fine-tuned to align with the attack characteristics of various attack methods.

To create the attack dataset, the principles of different attack methods are followed, and the appropriate attack messages are injected into the normal dataset at a certain frequency. We sequentially executed denial-of-service (DoS), fuzzing, replay, and spoofing attacks on the test set. Furthermore, each of these attack messages is labeled according to the type of attack to generate the attack dataset. The attack message m^{attack} associated with each attack exhibits the following characteristics:

DoS attack: This type of attack is designed to disrupt the normal operation of the system by consuming the bandwidth or resources of the CAN bus. The m^{attack} are characterized by having an ID field of 0x000 with all zero data fields.

Fuzzy Attack: This type of attack involves searching for vulnerabilities or anomalies within the system by sending specially crafted or randomly generated CAN-FD messages. The m^{attack} have random ID numbers ranging from 0×000 to 0×7FF, and their data fields are filled with random data.

Replay Attack: This type of attack imitates legitimate communication by recording or intercepting a previously legitimate message and then resending it to the bus. The m^{attack} are identical to the previous frame messages at their insertion locations.

Algorithm 2: Attack dataset generation

```
Input: Test dataset $\boldsymbol{M}_t$, total number of selected paragraphs $N_{attack}$, number of messages in each data segment $n^{attack}$, range of frames inserted $\left[k_{min}^{attack}, k_{max}^{attack}\right]$, attack message $m^{attack}$
Output: Datasets subjected to attack $\boldsymbol{M}_t^{attack}$.
$\boldsymbol{M}_t^{attack}$ = empty
$\left(\boldsymbol{M}_t^{attack,1}, \boldsymbol{M}_t^{attack,2}, \ldots, \boldsymbol{M}_t^{attack,N_{attack}}\right)$ ←select non-overlapping data segments $\left(\boldsymbol{M}_t, N_{attack}, p^{attack}\right)$
For $m_{t,i}$ in $\boldsymbol{M}_t$ do:
    append $m_{t,i} \rightarrow \boldsymbol{M}_t^{attack}$
    For $m_{t,i}$ in $(\boldsymbol{M}_t^{attack,1}, \boldsymbol{M}_t^{attack,2}, \ldots, \boldsymbol{M}_t^{attack,N_{attack}})$ do:
        $k^{attack} \leftarrow random\left(k_{min}^{attack}, k_{max}^{attack}\right)$
        For $k$ in range $\left(k^{attack}\right)$ do:
            append $m^{attack} \rightarrow \boldsymbol{M}_t^{attack}$
        End
    End
End
Return $\boldsymbol{M}_t^{attack}$
```

Spoofing Attack: In this type of attack, the vehicle's system is spoofed by sending forged CAN-FD data frames. To generate the m^{attack}, the message with a specific ID is selected from the test set, and a physical feature within the message is chosen. This feature has a start bit at b_{start} and an end bit at b_{end}. Prior to the insertion point of the m^{attack}, the nearest normal message m_{Benign} with the same ID is searched for, and the data within the range of $[b_{start}, b_{end}]$ of m_{Benign} is modified to create the m^{attack}. To ensure that the generated attack message closely simulates the scenario where an attacker performs a spoofing attack, it is crucial to make the modified feature value, $x_{Spoofing}$, sufficiently different from the original normal feature value, x_{Benign}. Therefore, the provision:

$$x_{Spoofing} = \begin{cases} x_{max} & x_{Benign} \leq \frac{1}{2}x_{max} \\ 0 & x_{Benign} > \frac{1}{2}x_{max} \end{cases} \tag{12}$$

To simulate the DoS attack scenario, six data segments, each containing 100 frames of messages, were randomly selected from the original test set. Subsequently, 0–6 frames of the m^{attack} were randomly inserted after each frame in these segments, as depicted in Figure 4a. To simulate the fuzzy attack scenario, six segments were randomly selected from the original test set, each containing 100 frames of messages. Subsequently, 0–3 frames of the m^{attack} were randomly inserted after each frame in every segment, as depicted in Figure 4b. To simulate the replay attack scenario, five data segments were randomly selected from the original test set, each containing 120 frames of messages. Furthermore, 0–2 frames of the m^{attack} were then randomly inserted after each frame in every segment, as illustrated in Figure 4c, to simulate the replay attack scenario. To simulate a spoofing attack scenario, five data segments were randomly selected from the original test set, each containing 120 frames. Moreover, 0–2 frames of m^{attack} were then randomly inserted after each frame in each segment to simulate the spoofing attack scenario. For example, a m^{attack} can be constructed by modifying bits [20,36] in the data field of the normal message with the ID of 0x010, which corresponds to the physical quantity of motor feedback torque. This simulates the scenario of a spoofing attack, as illustrated in Figure 4d.

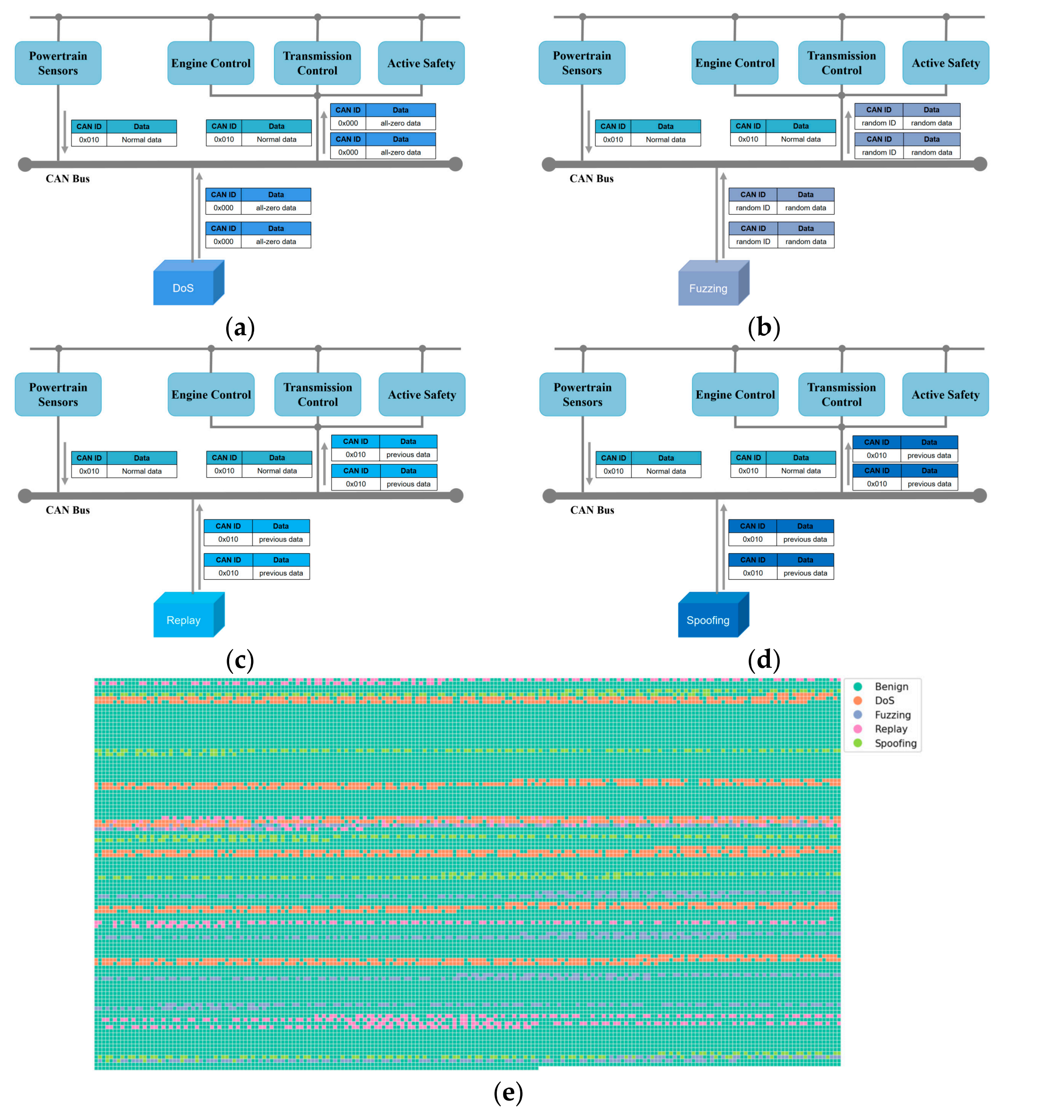


Figure 4. Generation of attack dataset: (**a**) DoS attack, (**b**) fuzzing attack, (**c**) replay attack, (**d**) spoofing attack, and (**e**) the distribution of attack messages.

4. Physical Semantics-Enhanced Anomaly Detection

4.1. Model Framework

To detect potential anomalous messages in CAN-FD datasets, an innovative CAN-FD anomaly detection method is introduced, grounded in the genuine physical interpretations of data fields. The schematic representation of this framework is visually depicted in Figure 5, comprising four integral components: extraction and standardization of features, model training, and anomaly evaluation. Initially, feature extraction involves the extraction of authentic physical meaning features pertinent to various ID messages within the dataset, adhering to predefined extraction rules for each ID message. Subsequently, feature standardization entails the expansion of feature counts for all ID messages to align with a standardized dimension denoted as q, where q signifies the maximum feature count value across all ID messages. The aforementioned features are subsequently restructured, consolidated, and normalized to correspond to the number of feature types within $\boldsymbol{M}_s$. The same feature extraction procedure is applied to each frame, yielding the concatenated input sequence $\boldsymbol{X}_{in}$ and output sequence $\boldsymbol{X}_{out}$. Model training is facilitated through the utilization of $\boldsymbol{X}_{in}$ and $\boldsymbol{X}_{out}$ for multiple self-attention mechanisms for time-series prediction model training. Anomaly evaluation requires the prediction of timing features using the trained multiple self-attention mechanisms for the time-series prediction model and determines whether an anomaly has occurred by comparing the difference between the predicted features and the actual features to determine whether the difference exceeds the threshold value.

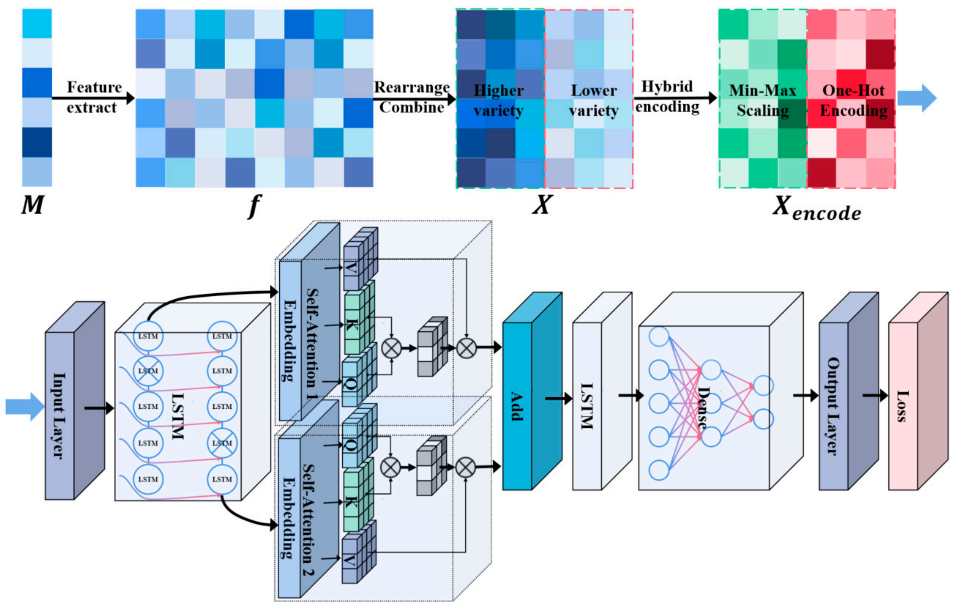
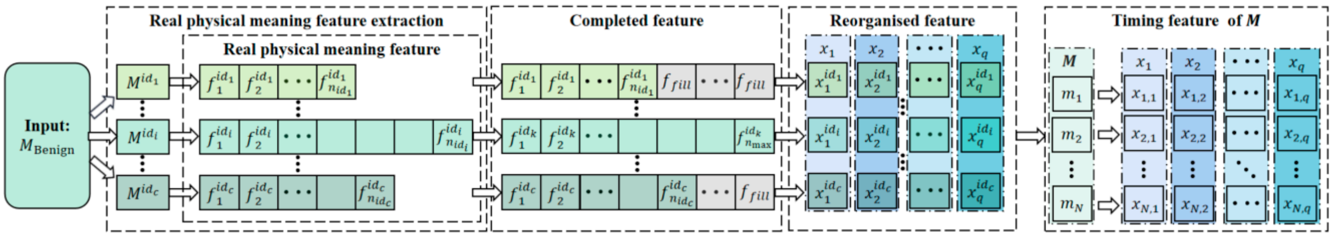


Figure 5. The CAN-FD anomaly detection framework based on genuine physical implications.

4.2. Extraction and Standardization of Features

The extraction and standardization of genuine physical meaning features were conducted on the training data, as illustrated in Figure 6, and are detailed in Algorithm 3. This algorithm necessitates initialization parameters, encompassing the training set denoted as $\boldsymbol{M}_s$, the rule base for extracting genuine physical meaning features from various ID messages represented as $\boldsymbol{\mathcal{R}}$, the filler feature f_{fill}, and the maximum number of genuine physical meaning features q. For each frame of messages within the training set $\boldsymbol{M}_s$, the genuine physical meaning features were extracted based on their corresponding ID, following the extraction rule $\boldsymbol{\mathcal{R}}$ to preserve dataset integrity. In cases where the number of extracted features fell below q, the feature count was augmented to q using the f_{fill}. Subsequently, the extracted features were organized in descending order, taking into account the number of data types for each feature within each ID. This arrangement effectively mitigated the dimensionality inflation resulting from subsequent selective thermal encoding. These rearranged features were also standardized. Finally, eigenvalues for features with constant values were aggregated to enhance the model's capacity to capture changing patterns in other critical features and to mitigate the risk of overfitting.



Input: M_{Benign}
Real physical meaning feature extraction
Real physical meaning feature
Completed feature
Reorganised feature
Timing feature of M
f_{fill}
$f_{n_{max}}^{id_k}$
x_1 x_2 x_q
m_1 m_2 m_N
$x_{1,1}$ $x_{1,2}$ $x_{1,q}$
$x_{N,1}$ $x_{N,2}$ $x_{N,q}$

Figure 6. The process of extracting genuine physical meaning features.

Algorithm 3: Feature extraction and standardization

```
Input: Training dataset $\boldsymbol{M}_s$, genuine physical meaning extraction rule base $\boldsymbol{\mathcal{R}}$, additional feature $f_{fill}$, maximum number of genuine physical meaning features $q$
Output: Standardized features $\boldsymbol{X}_s$.
$\boldsymbol{X}_s$ = empty, $\boldsymbol{X}_{s,combine}$ = empty
For $m_{s,i}$ in $\boldsymbol{M}_s$ do:
    $\boldsymbol{id} \leftarrow m_{s,i}$
    $\boldsymbol{F} \leftarrow extract\ feature(\boldsymbol{\mathcal{R}}, \boldsymbol{id}, m_{s,i})$
    If $len(\boldsymbol{F}) < q$ then:
        append $(q\text{-}len(\boldsymbol{F}))*f_{fill} \rightarrow \boldsymbol{F}$
    End
    append $\boldsymbol{F} \rightarrow \boldsymbol{X}_s$
End
$\boldsymbol{ID} \leftarrow count\ all\ ID\ types(\boldsymbol{M}_s)$
For $\boldsymbol{id}$ in $\boldsymbol{ID}$ do:
    $\boldsymbol{X}_s \leftarrow$ descend order by data type $(\boldsymbol{X}_s, \boldsymbol{id})$
    $\boldsymbol{X}_s \leftarrow$ standardized $(\boldsymbol{X}_s, \boldsymbol{id})$
End
For $\boldsymbol{X}_{s,i}$ in $\boldsymbol{X}_s$ do:
    If unique $(\boldsymbol{X}_{s,i})$==1 then:
        $\boldsymbol{X}_{s,combine} += \boldsymbol{X}_{s,i}$
        drop $\boldsymbol{X}_{s,i} \rightarrow \boldsymbol{X}_s$
    End
End
Return $\boldsymbol{X}_s$
```

To standardize the features, a min–max scaling approach is employed. For each feature type $\boldsymbol{X}$, its minimum value x_{min} and maximum value x_{max} are calculated. Subsequently, the feature values x within $\boldsymbol{X}$ are mapped to x_{nor} using the following equation:

$$x_{nor} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{13}$$

The features that are incorporated include the message IDs and their corresponding genuine physical meaning features. Features with a count of variable types not exceeding n_{encode} are encoded uniquely through a thermal encoding method, effectively mitigating the influence of weight bias and distance metrics. Conversely, for features with a variable type of count surpassing n_{encode}, their physical-value relationships are maintained to enrich the model's comprehension of their authentic physical meaning features.

It is essential to note that, in cases where a message lacks a corresponding extraction rule in $\boldsymbol{\mathcal{R}}$, its features will be populated with f_{fill}. During feature rearrangement in the test set $\boldsymbol{M}_t$, the order of rearrangement for each ID and the minimum x_{min} and maximum x_{max} for each feature type, used for standardization, remain consistent with those from the training set $\boldsymbol{M}_s$.

4.3. Training and Evaluation of Time-Series Prediction Models with Multiple Self-Attention Mechanisms

To obtain the input and output of the network model, features are extracted from all messages in the training set, which is denoted as $\boldsymbol{X}_s = (\boldsymbol{X}_{s,1}, \boldsymbol{X}_{s,2}, \ldots, \boldsymbol{X}_{s,q})$. Where $\boldsymbol{X}_{s,i} = (x_{1,i}, x_{2,i}, \ldots, x_{n_S,i})^T (i \in \{1, 2, \ldots, q\})$, indicating the *i*-th class of features extracted from the message. Meanwhile, $x_{j,i}$ ($j \in \{1, 2, \ldots, N_S\}$) represents a feature value extracted from the j-th frame of the message in the training set, and N_S denotes the total number of training samples. Therefore, for $\boldsymbol{X}_s$, the following relationship holds:

$$\boldsymbol{X}_s = \begin{bmatrix} \boldsymbol{X}_{s,1} \\ \boldsymbol{X}_{s,2} \\ \vdots \\ \boldsymbol{X}_{s,q} \end{bmatrix}^T = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,q} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,q} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_S,1} & x_{n_S,2} & \cdots & x_{n_S,q} \end{bmatrix} \tag{14}$$

The input sequence, denoted as $\boldsymbol{X}_{in}$, and the output sequence, denoted as $\boldsymbol{X}_{out}$, are composed of features extracted from distinct sliding windows, as depicted in Figure 7a:

$$\boldsymbol{X}_{in} = \left(\boldsymbol{X}_{in,1}, \boldsymbol{X}_{in,2}, \ldots, \boldsymbol{X}_{in,p}\right)^T \tag{15}$$

$$\boldsymbol{X}_{out} = \left(\boldsymbol{X}_{out,1}, \boldsymbol{X}_{out,2}, \ldots, \boldsymbol{X}_{out,p}\right)^T \tag{16}$$

where $\boldsymbol{X}_{in,i}$ and $\boldsymbol{X}_{out,i}$ ($i \in \{1, 2, \ldots, p\}$) form a paired relationship within the sequences $\boldsymbol{X}_{in}$ and $\boldsymbol{X}_{out}$. In other words, when $\boldsymbol{X}_{in,i}$ is provided as input, the anticipated output is $\boldsymbol{X}_{out,i}$ for which $\boldsymbol{X}_{in,i}$ and $\boldsymbol{X}_{out,i}$ have the following relationship:

$$\boldsymbol{X}_{in,i} = \begin{bmatrix} x_{1+iz,1} & x_{1+iz,2} & \cdots & x_{1+iz,q} \\ x_{2+iz,1} & x_{2+iz,2} & \cdots & x_{2+iz,q} \\ \vdots & \vdots & \ddots & \vdots \\ x_{u+iz,1} & x_{u+iz,2} & \cdots & x_{u+iz,q} \end{bmatrix} \tag{17}$$

$$\boldsymbol{X}_{out,i} = \begin{bmatrix} x_{u+1+iz,1} & x_{u+1+iz,2} & \cdots & x_{u+1+iz,q} \\ x_{u+2+iz,1} & x_{u+2+iz,2} & \cdots & x_{u+2+iz,q} \\ \vdots & \vdots & \ddots & \vdots \\ x_{u+v+iz,1} & x_{u+v+iz,2} & \cdots & x_{u+v+iz,q} \end{bmatrix} \tag{18}$$

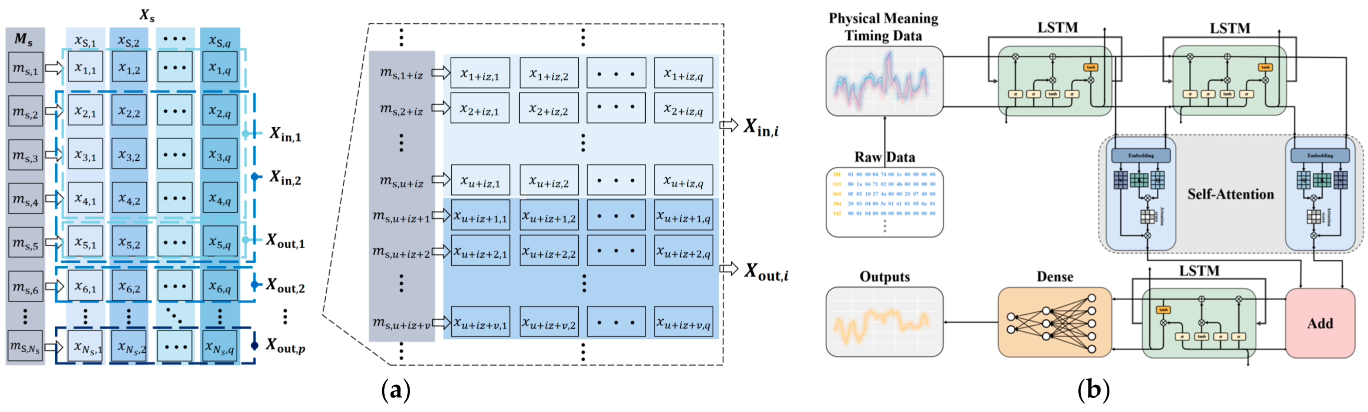


Figure 7. The training process of the model includes: (**a**) model inputs and outputs; (**b**) multiple self-attention mechanisms for time-series prediction model architecture.

The architecture of the multiple self-attention mechanisms for time-series prediction model, as depicted in Figure 7b, incorporates a two-layer LSTM network following the input layer. This addition is aimed at enhancing the model's ability to grasp temporal patterns and trends in the data, facilitating predictions for future time steps. Given that the cell states of the LSTM can transfer information across different time steps within the sequence, they are proficient at capturing long-term dependencies and prove effective in handling prediction tasks that demand the retention of extended historical information. LSTM's gating mechanism enables superior management of gradient flow, mitigating the issue of gradient vanishing or explosion that traditional RNNs often encounter when dealing with lengthy sequences. The primary components of the LSTM cell architecture include an input gate i_t, an output gate o_t, and a forget gate f_t. The LSTM relies on these three gates for feature selection, and their computation is as follows:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{19}$$

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right) \tag{20}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{21}$$

where W_i, W_f, and W_C represent the weighting matrices, b_i, b_f, and b_C are biases that are learned during the training process, x_t are input vectors, h_{t-1} are hidden input vectors, and σ signifies a sigmoid function. The process of generating candidate storage units $\overline{C_t}$ for updating the information is as follows:

$$\overline{C_t} = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{22}$$

Let C_t denote the new cell state and C_{t-1} represent the old cell state. The process of updating the cell state is as follows:

$$C_t = f_t C_{t-1} + i_t \overline{C_t} \tag{23}$$

The hidden output vector h_t is defined as:

$$h_t = o_t \tanh(C_t) \tag{24}$$

The forgetting gate f_t chooses the information to retain from the old cell state C_{t-1}, while the input gate o_t decides the amount of information to be preserved.

In order to facilitate the modeling of associations among different positions within the sequence, capturing long-range dependencies and crucial information, a self-attention layer is introduced after the two-layer LSTM network. The core concept of the self-attention mechanism is to project each element within the sequence into a high-dimensional space and then determine element-specific weights by assessing the similarity between different elements [38]. These weights signify the degree of interconnection between the elements. This correlation empowers the model to focus on significant information at various positions, thus enhancing the processing of sequence data.

To enhance the model's performance and bolster the network's expressive capacity, an Add layer for residual concatenation is integrated within the self-attention layer. Subsequently, an LSTM layer is incorporated to capture and analyze temporal characteristics and patterns within the dataset. The final stage entails feature extraction and transformation through a fully connected layer, resulting in the generation of the output, denoted as $\boldsymbol{X}_{out}$.

5. Performance Evaluation and Comparative Analysis

5.1. Experiment Setup

Our experiments were conducted using Python 3.10 in the PyCharm IDE version 2023.1.2, and we implemented Keras 2.13.1 with TensorFlow 2.13.0 as the backend on a computer system equipped with an AMD Ryzen 7925HX with Radeon Graphics CPU, operating at 2.50 GHz, and 16 GB of RAM. The operating environment was Windows 11 (64-bit), and the system was equipped with an NVIDIA GeForce GTX 4060 laptop GPU.

The sliding window size for the input sequence was configured as $u = 4$, and the size of the output sequence was set to =1. After data preprocessing, the relevant input features and desired outputs were extracted from the training set. Subsequently, the training of the LSTM network model, combined with the attention mechanism, was carried out. The parameters of the network model were set in accordance with the specifications outlined in Table 1.

Table 1. The configuration of model parameters.

Hyperparameter	**Value**
Learning Rate	10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}
Epoch Number	100
Batch Size	16, 32, 64, 128, 256
LSTM Unit Number	512, 256, 64
Self-Attention Unit Number	128
Dropout	0.2
Recurrent Dropout	0.2
Loss Function	MSE, MAE, MSLE, Huber, Log Cosh
Activation	ReLU, Sigmoid, Tanh, Leaky Relu, ELU, SeLU
Number of attentions	0, 12, 3

CAN-FD datasets were procured from real vehicles, as illustrated in Figure 8. The CAN-FD transceiver, Kvaser HybridPro 2XCAN/LIN (manufactured by Kvaser, based in Brumma, Sweden), was employed to receive each CAN-FD message from the vehicle's powertrain subnetwork interface. Subsequently, it facilitated the conversion of these messages into information, including ID, timestamp, and data fields, which were then uploaded to a PC server. On the PC side, the data fields of the CAN-FD message for each ID are correlated with the physical meaning of the characterization application through the utilization of the CAN-FD application layer protocol matrix supplied by the vehicle manufacturer. The original format of this dataset is in hexadecimal and comprises a total of 84,956 frames. Eighty percent of the messages in this dataset were allocated to the training set, with the remaining 20% reserved for the test set. The training set encompasses 67,964 frames, while the test set comprises 16,992 frames. Subsequently, various types of attack messages were introduced into the training set to create the attack dataset, and the distinct message types were appropriately labeled. Among the novel classification labels introduced in the dataset, m_{Benign} signifies normal messages, m_{DoS} represents DoS attack messages, m_{Fuzzy} characterizes fuzzy attack messages, m_{Replay} designates replay attack messages, and $m_{Spoofing}$ denotes spoofing attack messages. The newly generated CAN-FD attack dataset comprises a total of 20,928 frames, with the distribution of attack messages illustrated in Figure 4e. Within this dataset, there are 1789 frames containing DoS attack messages, 837 frames containing fuzzing attack messages, 571 frames featuring replay attack messages, and 607 frames containing spoofing attack messages. As indicated in Table 2, the proportion of normal messages in the generated attack dataset stands at 81.71%, with DoS attack messages constituting 8.60%, fuzzing attack messages comprising 4.02%, replay attack messages accounting for 2.75%, and spoofing attack messages representing 2.92%.

These datasets were segregated in accordance with the concrete physical interpretations of data fields associated with various IDs. This partitioning facilitated the extraction of data segments with discernible physical connotations, which were subsequently transmuted into the corresponding real-world physical significance eigenvalues. The initial set of IDs encompassed 0x010, 0x142, 0x060, 0x304, and 0x18f.

The data field associated with the message ID 0x010 denoted physical quantities linked to MCU torque feedback. Message ID 0x142 was indicative of DBS state-related quantities, while message ID 0x060 encapsulated physical quantities associated with MCU drive motor feedback. Moreover, message ID 0x304 featured a data field representing the physical quantity of VCU vehicle state 2, and message ID 0x18f contained a data field representing the physical quantity associated with EPS state.

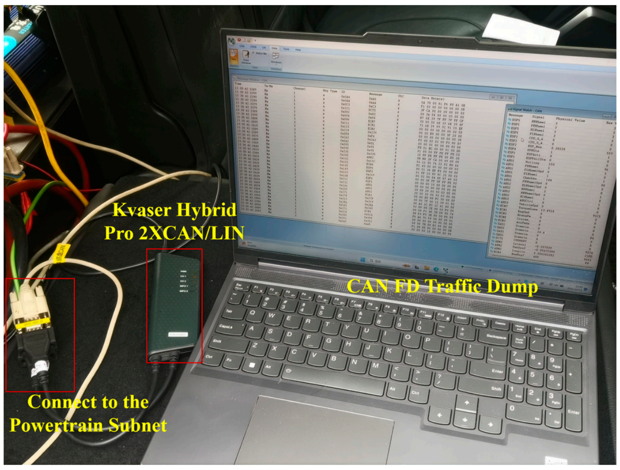


Figure 8. Collection of authentic CAN-FD datasets from real vehicles.

Table 2. The types and corresponding percentages of messages within the attack dataset.

Message Type	Number of Instances	Percentage of Instances
m_{Benign}	16,992	81.71%
m_{DoS}	1789	8.60%
m_{Fuzzy}	837	4.02%
m_{Replay}	571	2.75%
$m_{Spoofing}$	607	2.92%

All data segments with genuine physical meanings embedded within these data fields were extracted. The values within these segments were converted to decimal format, categorized in accordance with Table 3, and assigned the corresponding feature labels in a sequential order. In order to ensure consistency, "-1" was employed to augment the number of features within each ID message, aligning them with the maximum number of features as specified in Table 3.

Table 3. The genuine physical content encapsulated within the data fields corresponding to different IDs.

ID	0x010	0x142	0x060	0x304	0x18f
Message	MCU Torque Feedback	DBS Status	MCU Drive Motor Feedback	VCU Vehicle Status 2	EPS Status
Feature 1	MCU Shift	DBS System Status	Motor Udc	Vehicle Speed	EPS Work Mode
Feature 2	MCU Speed	DBS Park Warning	Motor ldc	Vehicle Brake Pressure	EPS Fault
Feature 3	MCU Torque	DBS Work Mode	Motor Control Temp	Vehicle Steering Angle	EPS Calibration Status
Feature 4	MCU Current	Brake Pressure Reg ACK	MCU Motor Error Grade	-	EPS Str Angle Act
Feature 5	MCU Motor temp	DBS HP Pressure	Clamping Brake Status	-	EPS Motor Current
Feature 6	MCU Error Code	DBS Peadal Opening	-	-	EPS Temperature
Feature 7	-	DBS Ref Iq	-	-	-
Feature 8	-	DBS Rolling Counter	-	-	-
Feature 9	-	DBS Estop Flag	-	-	-
Feature 10	-	DBS Pedai Flag	-	-	-
Feature 11	-	DBS Check Sum	-	-	-

Additionally, the features within each ID message were sorted in descending order based on the nature of their values. Concurrently, they underwent normalization through min-max scaling. Finally, feature merging was carried out on features that maintained unchanged values.

5.2. Performance Evaluation

5.2.1. Evaluation Metrics

The trained model is validated through the utilization of the input features from the attack dataset in conjunction with their corresponding actual outputs. The evaluation of the model's performance relies on a set of metrics, including TP (true positives), FP (false positives), TN (true negatives), FN (false negatives), accuracy, precision, recall, and F1 score. In order to assess the model's performance, the trained model is loaded, and predictions are made for each element in the input features of the attack dataset. For each prediction, the mean squared loss value is computed, comparing the predicted value with its corresponding actual output.

A mean squared error threshold θ is established. If the mean square error loss value falls below θ, the prediction result aligns with our expectations, and the message is classified as normal. Conversely, if the mean square error loss value exceeds θ, the prediction result deviates from our expectations, and the message is deemed abnormal. The definitions of TP, FP, TN, and FN are as follows:

- **TP**: Represents instances where the model correctly identifies positive cases as positive.
- **FP**: Denotes cases where the model incorrectly identifies negative instances as positive.
- **TN**: Corresponds to cases where the model accurately identifies negative instances as negative.
- **FN**: Refers to instances where the model mistakenly identifies positive cases as negative.

Accuracy is calculated as the ratio of correctly predicted samples to the total number of samples in the model. A higher accuracy value indicates superior model performance. The accuracy can be calculated using the following formula:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{25}$$

Precision measures the proportion of correctly predicted positive cases among all samples that were predicted to be positive by the model. A high precision rate signifies a high degree of accuracy in the model's positive case predictions. It can be calculated using the following formula:

$$Precision = \frac{TP}{TP + FP} \tag{26}$$

Recall measures the proportion of all actual positive examples that the model correctly predicts as positive. A high recall indicates the model's proficiency in recognizing positive examples. It is calculated as follows:

$$Recall = \frac{TP}{TP + FN} \tag{27}$$

The F1 score is a metric that harmonizes precision and recall, providing a more comprehensive evaluation of the model's performance. It is calculated as follows:

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{28}$$

5.2.2. Experimental Verification

To obtain a lucid assessment of the model's performance, it is imperative to evaluate the model's efficacy using the aforementioned performance metrics. Figure 9 graphically depicts the relationship between the model's predictions and the actual observations through the use of a confusion matrix, especially the model's predictive accuracy and the rate of misclassification for normal and anomalous messages.

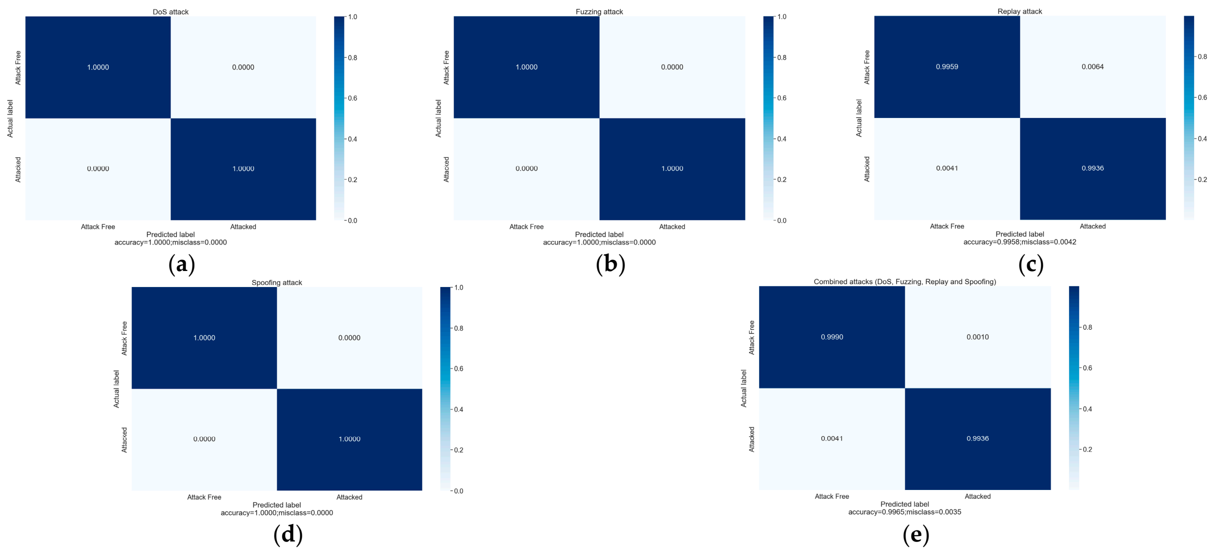


Figure 9. The model's performance in recognizing various types of attacks: (**a**) DoS attacks, (**b**) fuzzing attacks, (**c**) replay attacks, (**d**) spoofing attacks, and (**e**) hybrid attacks.

Leveraging the incorporation of authentic physical meanings in the data fields as input features and the utilization of an LSTM-based network model combined with a multi-head self-attention mechanism, the model accurately anticipates changes in the temporal characteristics of the genuine physical meanings. Table 4 illustrates the model's accuracy in identifying different types of attacks. Under the optimal hyper-parameter configuration, the model achieves a remarkable accuracy of 99.646% in identifying anomalous messages, with an impressive F1 score of 0.99782. The recognition accuracy for DoS, fuzzing, and spoofing attacks reaches 100%, while the recognition accuracy for replay attacks, which are typically challenging to discern, remains notably high at 99.36%.

Table 4. The model's accuracy in recognizing a multitude of attack types.

Type	$m_{\mathbf{DoS}}$	$m_{\mathbf{Fuzzy}}$	$m_{\mathbf{Replay}}$	$m_{\mathbf{Spoofing}}$
TN	1813	864	624	622
FN	0	0	4	0
Accuracy	100%	100%	99.36%	100%

To ascertain the significance of learning features with genuine physical meaning in messages and their impact on enhancing the model's anomaly detection capability, it is imperative to evaluate the model's performance when learning different input features. Reference [39] employs a deep learning-based LSTM algorithm for the detection of timing data anomalies in CAN networks. This approach demonstrates a combination of a low false alarm rate and a high detection rate, leveraging message data field byte features as input at the data link layer level. The approach in this paper, employing application layer physical semantics as input, was compared with the approach from the literature [39], which employs data link layer byte features as input, under identical training and test sets. Following the approach presented in [39], each byte (every 4 bits) within the data field was treated as a feature. In addition to this, the model's performance was assessed when considering every two bytes (every 8 bits) as input features.

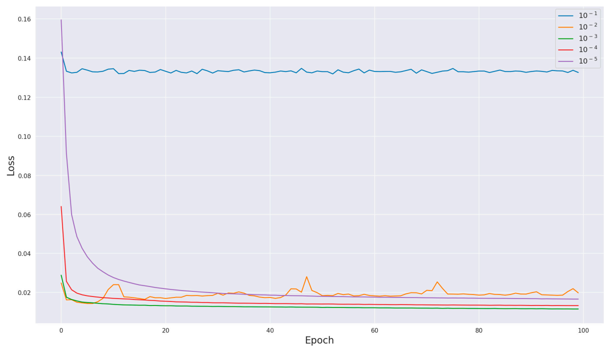
The specific outcomes are detailed in Table 5. Remarkably, models that employ input features derived from the genuine physical interpretation of data fields exhibit a 4–5% enhancement in anomaly detection accuracy when contrasted with non-physical content-aware approaches. Meanwhile, our method exhibits a misclassification rate of merely 0.412% for normal messages and a mere 0.102% for anomalous messages, whereas the method introduced in [39] demonstrates a misclassification rate exceeding 2.4% for normal messages and surpassing 12.9% for anomalous messages. This observation underscores the effectiveness of employing application-layer physical interpretations in anomaly detection, enabling the differentiation of fluctuations arising from typical operations from genuine anomalies and thereby substantially diminishing the misclassification rate for normal messages.

Table 5. Comparison of model performance when learning different input features.

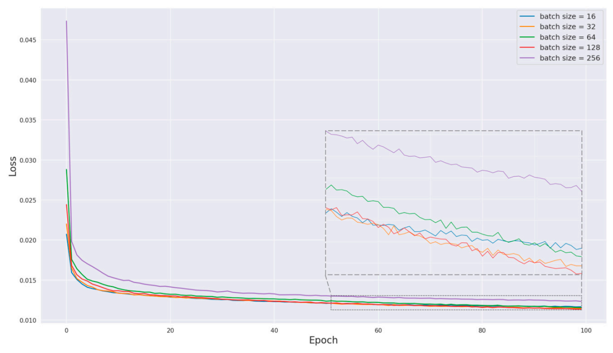
Input Features	TP	FP	TN	FN	Accuracy	Precision	Recall	F1 Score
Methodology of this paper	16,918	70	3923	4	99.646%	99.588%	99.976%	0.99782
Each byte [39]	16,395	593	3420	507	94.741%	96.509%	97.000%	0.96754
Every two bytes [39]	16,580	408	411	516	95.582%	97.589%	96.982%	0.97289

5.2.3. Ablation Study

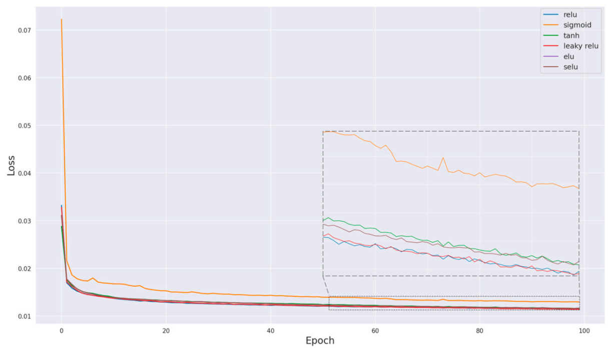
To determine the optimal learning rate, batch size, and activation function for further enhancing model performance, a comprehensive analysis was conducted. The model's performance was assessed by comparing loss curves obtained under varying hyperparameters. Figure 10a displays the loss curves derived from training the model with different learning rates. Notably, when the learning rate is set at 0.001, the model exhibits a consistent decline in loss values throughout the training epochs, ultimately yielding the lowest loss value, thus yielding the most favorable training outcome. Subsequently, using this optimal learning rate, the impact of different batch sizes on the training performance was evaluated, as depicted in Figure 10b. The most favorable training outcome was achieved with a batch size of 128. Following this, the effect of various activation functions on model training was explored, as illustrated in Figure 10c, where it was observed that the leaky ReLU activation function demonstrated superior performance.



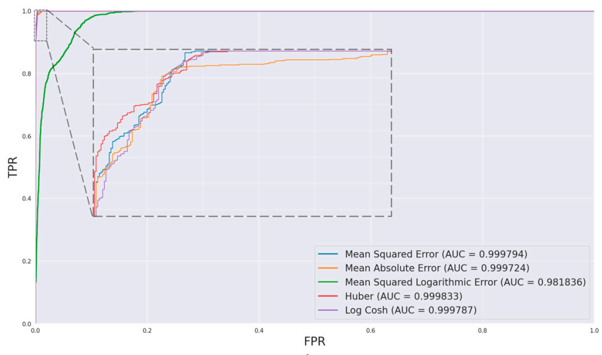
(**a**)



(**b**)



(**c**)



(**d**)

Figure 10. Evaluation of different hyperparameters on model performance: (**a**) learning rate, (**b**) batch size, (**c**) activation function, and (**d**) ROC curves for different loss functions.

Furthermore, multiple loss functions were considered, including mean squared error, mean absolute error, mean squared logarithmic error, Huber, and log cosine error. The formula for mean squared error is provided below:

$$MSE = \frac{1}{n}\sum\nolimits_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{29}$$

where n is the number of samples, y_i is the true value of the i-th sample, and $\hat{y}_i$ is the predicted value of the i-th sample.

The formula for the mean absolute error is as follows:

$$MAE = \frac{1}{n}\sum\nolimits_{i=1}^{n}|y_i - \hat{y}_i| \tag{30}$$

The formula for the mean squared logarithmic error is expressed as follows:

$$MSLE = \frac{1}{n}\sum\nolimits_{i=1}^{n}(\log(y_i) - \log(\hat{y}_i))^2 \tag{31}$$

The Huber loss function is defined by the following formula:

$$Huber = \frac{1}{n}\sum\nolimits_{i=1}^{n}\begin{cases} (y_i - \hat{y}_i)^2 |y_i - \hat{y}_i| \le \delta \\ -\frac{1}{2}\delta^2 + \delta|y_i - \hat{y}_i||y_i - \hat{y}_i| > \delta \end{cases} \tag{32}$$

where δ is a non-negative constant.

The formula for the logarithmic cosine error can be expressed as follows:

$$\text{Logcos}h = \frac{1}{n}\sum\nolimits_{i=1}^{n}\log\left(e^{y_i - \hat{y}_i} + e^{\hat{y}_i - y_i}\right) \tag{33}$$

The performance of the aforementioned five loss functions on the attack dataset is detailed in Table 6, while their corresponding receiver operating characteristic (ROC) curves are presented in Figure 10d. With the exception of MSLE, the area under the curve (AUC) scores of the other loss functions exhibit proximity to one another. Among these, Huber demonstrates a remarkable level of accuracy and F1 score in recognizing attack messages, second only to MSE, while concurrently achieving the highest AUC score. Consequently, it is concluded that Huber is the optimal choice for the loss function.

Table 6. The impact of various loss functions on model performance.

Loss Function	TP	FP	TN	FN	Accuracy	Precision	Recall	F1 Score
Mean Squared Error	16,925	63	3925	2	99.689%	99.629%	99.988%	0.99808
Mean Absolute Error	16,805	183	3926	1	99.120%	98.923%	99.994%	0.99456
Mean Squared Logarithmic Error	15,296	1692	3843	84	91.508%	90.040%	99.454%	0.94513
Huber	16,918	70	3923	4	99.646%	99.588%	99.976%	0.99782
Log Cosh	16,913	75	3926	1	99.637%	99.559%	99.994%	0.99776

To assess the effectiveness of the self-attention mechanism in enhancing the model's ability to capture temporal features within the input data, models with variations were developed. These variations included models devoid of the attention mechanism and models employing different quantities of self-attention heads. The performance of these models is presented in Table 7. Notably, the most superior AUC score and overall performance were attained when utilizing two self-attention heads.

Table 7. Comparison of model performance with an equal number of self-attention heads.

Number of Attention	Accuracy	Precision	Recall	F1 Score	AUC
None	99.627%	99.546%	99.994%	0.99770	0.999811
One	99.574%	98.488%	99.988%	0.99737	0.999790
Two	99.646%	99.588%	99.976%	0.99782	0.999833
Three	99.603%	99.529%	99.982%	0.99755	0.999805

6. Conclusions

In this paper, an anomaly detection method for CAN-FD networks is proposed, with a focus on ascertaining the presence of abnormal network traffic by analyzing temporal changes in the genuine physical semantics within message data fields. The method commences by initially gathering CAN-FD datasets obtained from real vehicles devoid of any cyberattacks. These datasets are subsequently partitioned into a training set and a test set. The training set is further enriched through the adjustment of the sliding window step size and the introduction of sequence order disruptions. Simultaneously, the attack dataset is formulated by infusing distinct categories of attack messages into the test set, each with varying frequencies based on their specific attack attributes. Genuine physical semantic features within the message data fields are extracted and subsequently subjected to preprocessing, encompassing tasks such as padding, reordering, normalization, and consolidation. Following this, a hybrid coding methodology is applied to effectively mitigate the dimensionality inflation that typically arises from one-hot encoding. The multiple self-attention mechanisms for time-series prediction model architecture integrates the LSTM network and incorporates a multi-head self-attention mechanism to capture temporal features in the training set. The model's hyperparameters are meticulously fine-tuned to optimize its performance. The model's performance is then validated using the attack dataset, and its performance under varying input features is assessed. The results reveal a substantial enhancement in anomaly detection accuracy, approximately 4–5%, when utilizing the genuine physical semantics of message data fields as inputs, compared to the utilization of byte features at the data link layer level. The proposed method proves highly effective in detecting various attack types, including DoS, fuzzing, replay, and spoofing, achieving exceptional performance metrics with an F1 score of 0.99782 and an accuracy of 99.646%. Future research will delve into anomaly detection in vehicular networks, leveraging graph theory and the incorporation of the genuine physical semantics of messages.

Author Contributions: Conceptualization, R.Z. and C.L.; methodology, R.Z.; software, C.L.; validation, R.Z., C.L. and F.G.; formal analysis, C.L.; investigation, D.Z.; resources, Z.G.; data curation, L.L. and W.Y.; writing—original draft preparation, C.L.; writing—review and editing, F.G.; visualization, C.L.; supervision, R.Z.; project administration, F.G.; funding acquisition, Z.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under Grants 52202494 and 52202495.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to the need for confidentiality of application layer protocols for car companies.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lin, C.W.; Sangiovanni-Vincentelli, A. Cyber-security for the controller area network (CAN) communication protocol. In Proceedings of the 2012 International Conference on Cyber Security, Alexandria, VA, USA, 14–16 December 2012; pp. 1–7.
2. Lee, H.; Jeong, S.H.; Kim, H.K. OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame. In Proceedings of the 2017 15th Annual Conference on Privacy, Security and Trust (PST), Calgary, AB, Canada, 28–30 August 2017; pp. 57–5709.

3. Islam, R.; Refat, R.U.D. Improving CAN bus security by assigning dynamic arbitration IDs. *J. Transp. Secur.* **2020**, *13*, 19–31. [CrossRef]
4. Koscher, K.; Czeskis, A.; Roesner, F.; Patel, S.; Kohno, T.; Checkoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H.; et al. Experimental security analysis of a modern automobile. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 16–19 May 2010; pp. 447–462.
5. Greenberg, A. Hackers remotely kill a jeep on the highway—With me in it. *Wired* **2015**, *7*, 21–22.
6. Iehira, K.; Inoue, H.; Ishida, K. Spoofing attack using bus-off attacks against a specific ECU of the CAN bus. In Proceedings of the 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 12–15 January 2018; pp. 1–4.
7. Farag, W.A. CANTrack: Enhancing automotive CAN bus security using intuitive encryption algorithms. In Proceedings of the 2017 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO), Sharjah, United Arab Emirates, 4–6 April 2017; pp. 1–5.
8. Jo, H.J.; Kim, J.H.; Choi, H.Y.; Choi, W.; Lee, D.H.; Lee, I. Mauth-can: Masquerade-attack-proof authentication for in-vehicle networks. *IEEE Trans. Veh. Technol.* **2019**, *69*, 2204–2218. [CrossRef]
9. Xie, G.; Yang, L.T.; Wu, W.; Zeng, K.; Xiao, X.; Li, R. Security enhancement for real-time parallel in-vehicle applications by CAN FD message authentication. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 5038–5049. [CrossRef]
10. Xie, G.; Yang, L.T.; Liu, Y.; Luo, H.; Peng, X.; Li, R. Security enhancement for real-time independent in-vehicle CAN-FD messages in vehicular networks. *IEEE Trans. Veh. Technol.* **2021**, *70*, 5244–5253. [CrossRef]
11. Moore, M.R.; Bridges, R.A.; Combs, F.L.; Starr, M.S.; Prowell, S.J. Modeling inter-signal arrival times for accurate detection of can bus signal injection attacks: A data-driven approach to in-vehicle intrusion detection. In Proceedings of the 12th Annual Conference on Cyber and Information Security Research, Oak Ridge, TN, USA, 4–6 April 2017; pp. 1–4.
12. Kuwahara, T.; Baba, Y.; Kashima, H.; Kishikawa, T.; Tsurumi, J.; Haga, T.; Ujiie, Y.; Sasaki, T.; Matsushima, H. Supervised and unsupervised intrusion detection based on CAN message frequencies for in-vehicle network. *J. Inf. Process.* **2018**, *26*, 306–313. [CrossRef]
13. Song, H.M.; Kim, H.R.; Kim, H.K. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In Proceedings of the 2016 International Conference on Information Networking (ICOIN), Kota Kinabalu, Malaysia, 13–15 January 2016; pp. 63–68.
14. Salem, M.; Crowley, M.; Fischmeister, S. Anomaly detection using inter-arrival curves for real-time systems. In Proceedings of the 2016 28th Euromicro Conference on Real-Time Systems (ECRTS), Toulouse, France, 5–8 July 2016; pp. 97–106.
15. Müter, M.; Asaj, N. Entropy-based anomaly detection for in-vehicle networks. In Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, 5–9 June 2011; pp. 1110–1115.
16. Marchetti, M.; Stabili, D.; Guido, A.; Colajanni, M. Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms. In Proceedings of the 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI), Bologna, Italy, 7–9 September 2016; pp. 1–6.
17. Cho, K.T.; Shin, K.G. Fingerprinting electronic control units for vehicle intrusion detection. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; pp. 911–927.
18. Ji, H.; Wang, Y.; Qin, H.; Wu, X.; Yu, G. Investigating the effects of attack detection for in-vehicle networks based on clock drift of ECUs. *IEEE Access* **2018**, *6*, 49375–49384. [CrossRef]
19. Halder, S.; Conti, M.; Das, S.K. COIDS: A clock offset based intrusion detection system for controller area networks. In Proceedings of the 21st International Conference on Distributed Computing and Networking, Kolkata, India, 4–7 January 2020; pp. 1–10.
20. Choi, W.; Joo, K.; Jo, H.J.; Park, M.C.; Lee, D.H. VoltageIDS: Low-level communication characteristics for automotive intrusion detection system. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 2114–2129. [CrossRef]
21. Levy, E.; Shabtai, A.; Groza, B.; Murvay, P.S.; Elovici, Y. CAN-LOC: Spoofing detection and physical intrusion localization on an in-vehicle CAN bus based on deep features of voltage signals. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 4800–4814. [CrossRef]
22. Yin, L.; Xu, J.; Wang, C.; Wang, Q.; Zhou, F. Detecting CAN overlapped voltage attacks with an improved voltage-based in-vehicle intrusion detection system. *J. Syst. Archit.* **2023**, *143*, 102957. [CrossRef]
23. Olufowobi, H.; Young, C.; Zambreno, J.; Bloom, G. Saiducant: Specification-based automotive intrusion detection using controller area network (can) timing. *IEEE Trans. Veh. Technol.* **2019**, *69*, 1484–1494. [CrossRef]
24. Marchetti, M.; Stabili, D. Anomaly detection of CAN bus messages through analysis of ID sequences. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 1577–1583.
25. Yu, T.; Wang, X. Topology verification enabled intrusion detection for in-vehicle CAN-FD networks. *IEEE Commun. Lett.* **2019**, *24*, 227–230. [CrossRef]
26. Wang, C.; Zhao, Z.; Gong, L.; Zhu, L.; Liu, Z.; Cheng, X. A distributed anomaly detection system for in-vehicle network using HTM. *IEEE Access* **2018**, *6*, 9091–9098. [CrossRef]
27. Taylor, A.; Leblanc, S.; Japkowicz, N. Anomaly detection in automobile control network data with long short-term memory networks. In Proceedings of the 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, Canada, 17–19 October 2016; pp. 130–139.
28. Dong, C.; Wu, H.; Li, Q. Multiple Observation HMM-based CAN bus Intrusion Detection System for In-Vehicle Network. *IEEE Access* **2023**, *11*, 35639–35648. [CrossRef]

29. Zhang, H.; Zeng, K.; Lin, S. Federated graph neural network for fast anomaly detection in controller area networks. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 1566–1579. [CrossRef]
30. Xie, Y.; Zeng, G.; Ryo, K.; Xie, G.; Dou, Y.; Zhou, Z. An optimized design of CAN-FD for automotive cyber-physical systems. *J. Syst. Archit.* **2017**, *81*, 101–111. [CrossRef]
31. Zago, G.M.; de Freitas, E.P. A quantitative performance study on CAN and CAN-FD vehicular networks. *IEEE Trans. Ind. Electron.* **2017**, *65*, 4413–4422. [CrossRef]
32. Lee, Y.; Woo, S. CAN Signal Extinction-based DoS Attack on In-Vehicle Network. *Secur. Commun. Netw.* **2022**, *2022*, 9569703. [CrossRef]
33. Lee, H.; Choi, K.; Chung, K.; Kim, J.; Yim, K. Fuzzing can packets into automobiles. In Proceedings of the 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, Gwangju, Republic of Korea, 24–27 March 2015; pp. 817–821.
34. Naha, A.; Teixeira, A.; Ahlén, A.; Dey, S. Sequential detection of replay attacks. *IEEE Trans. Autom. Control* **2022**, *68*, 1941–1948. [CrossRef]
35. Yang, Y.; Duan, Z.; Tehranipoor, M. Identify a spoofing attack on an in-vehicle CAN bus based on the deep features of an ECU fingerprint signal. *Smart Cities* **2020**, *3*, 17–30. [CrossRef]
36. Arantes, R.B.; Vogiatzis, G.; Faria, D.R. Learning an augmentation strategy for sparse datasets. *Image Vis. Comput.* **2022**, *117*, 104338. [CrossRef]
37. Cubuk, E.D.; Zoph, B.; Mane, D.; Vasudevan, V.; Le, Q.V. Autoaugment: Learning augmentation strategies from data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 113–123.
38. Qiu, D.; Yang, B. Text summarization based on multi-head self-attention mechanism and pointer network. *Complex Intell. Syst.* **2022**, *8*, 555–567. [CrossRef]
39. Qin, H.; Yan, M.; Ji, H. Application of controller area network (CAN) bus anomaly detection based on time series prediction. *Veh. Commun.* **2021**, *27*, 100291. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.