CANShield: Deep-Learning-Based Intrusion Detection Framework for Controller Area Networks at the Signal Level

Md Hasan Shahriar, *Student Member, IEEE*, Yang Xiao, *Member, IEEE*, Pablo Moriano, *Senior Member, IEEE*, Wenjing Lou, *Fellow, IEEE*, and Y. Thomas Hou, *Fellow, IEEE*

***Abstract*—Modern vehicles rely on a fleet of electronic control units (ECUs) connected through controller area network (CAN) buses for critical vehicular control. With the expansion of advanced connectivity features in automobiles and the elevated risks of internal system exposure, the CAN bus is increasingly prone to intrusions and injection attacks. As ordinary injection attacks disrupt the typical timing properties of the CAN data stream, rule-based intrusion detection systems (IDS) can easily detect them. However, advanced attackers can inject false data to the signal/semantic level, while looking innocuous by the pattern/frequency of the CAN messages. The rule-based IDS, as well as the anomaly-based IDS, are built merely on the sequence of CAN messages IDs or just the binary payload data and are less effective in detecting such attacks. Therefore, to detect such intelligent attacks, we propose CANShield, a deep learning-based signal-level intrusion detection framework for the CAN bus. CANShield consists of three modules: 1) a data preprocessing module that handles the high-dimensional CAN data stream at the signal level and parses them into time series suitable for a deep learning model; 2) a data analyzer module consisting of multiple deep autoencoder (AE) networks, each analyzing the time-series data from a different temporal scale and granularity; and 3) finally an attack detection module that uses an ensemble method to make the final decision. Evaluation results on two high-fidelity signal-based CAN attack data sets show the high accuracy and responsiveness of CANShield in detecting advanced intrusion attacks.**

***Index Terms*—Controller area networks (CANs), deep learning, ensemble method, intrusion detection systems (IDS).**

Manuscript received 9 May 2023; revised 1 July 2023; accepted 26 July 2023. Date of publication 8 August 2023; date of current version 7 December 2023. This work was supported in part by the U.S. National Science Foundation (NSF) under Grant CNS-1837519 and Grant CNS-2235232; in part by the Virginia Commonwealth Cyber Initiative (CCI); and in part by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL), managed by UT-Battelle, LLC, for the U.S. Department of Energy. *(Corresponding author: Md Hasan Shahriar.)*

Md Hasan Shahriar and Wenjing Lou are with the Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 USA (e-mail: hshahriar@vt.edu; wjlou@vt.edu).

Yang Xiao is with the Department of Computer Science, University of Kentucky, Lexington, KY 40506 USA (e-mail: xiaoy@uky.edu).

Pablo Moriano is with the Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37930 USA (e-mail: moriano@ornl.gov).

Y. Thomas Hou is with the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 USA (e-mail: thou@vt.edu).

Digital Object Identifier 10.1109/JIOT.2023.3303271

I. Introduction

MODERN vehicles are becoming increasingly computerized to ensure driver's safety and convenience. The fusion of multimodal data from different types of sensors enables vehicles to recognize the driving context and make crucial decisions. The majority of the vehicles' critical functionalities, including acceleration, braking, steering, engine control, etc., involve dedicated microcontroller modules, known as electronic control units (ECUs), which are connected by one or more automotive communication buses running standardized protocols. Controller area network (CAN), also known as the CAN bus protocol, is the de facto automobile communication standard for safety-critical ECUs [1]. More recently, CAN bus enables vehicles to implement advanced driver assistance systems (ADAS), one of the fastest-growing applications in the automotive sector, providing enhanced passenger experience and safety. Moreover, advancements in wireless communication technology (e.g., 5G and V2X) have enabled the interface to connect with the internal ECUs from the outside network to conduct diagnostics or update firmware over-the-air (FOTA) remotely, rather than visiting a service facility [2]. Infotainment features, such as Bluetooth, Wi-Fi, and other smart interfaces, are also becoming prevalent in automobiles to add more convenience to the passengers [1]. Besides, the integration of Internet of Things (IoT) technology in the automotive industry, also known as Automotive IoT presents huge opportunities [3], such as optimizing the vehicles' performance, improving transportation management, and enhancing vehicle safety through predictive maintenance, AI-powered driving assistance, connectivity, etc.

The increased connectivity of modern vehicles as well as Automotive IoT technologies nonetheless increases the susceptibility of vehicular systems to remote attacks and message injections. The ability to hijack an ECU and inject stealthy messages into the vehicles' internal communication systems allows attackers to circumvent a wide array of safety-critical systems and control a wide range of vehicular functions. Researchers discovered several remote access points on connected vehicles and demonstrated that attackers could remotely exploit them to take control of the vehicles, including disabling the brakes, braking individual wheels, stopping the engine, and so on [4], [5]. For instance, Miller [6] remotely compromised a Jeep and transmitted malicious CAN messages, which led to the vehicle malfunctioning on the highway. Later, Chrysler recalled 1.4 million vehicles that can be remotely hacked over the Internet [7].

Despite the CAN protocol's widespread implementation and high reliability, it remains vulnerable to intruders due to the absence of basic security mechanisms as they introduce delays in message transmission or increase bus traffic [8]. Although there are a few works on implementing message authentication code (MAC) on the CAN bus to authenticate the sender ECU and prevent different attacks, they are costly and only achieve limited cryptographic strength [9], [10]. Moreover, it is difficult to insert the MAC along with the CAN message because of the limited payload length. As a result, only the plaintext message is broadcast over the CAN bus. Hence, CAN protocol does not include a way to verify where the message comes from or its integrity [8]. Due to this security deficiency, vehicles using the CAN protocol remains insecure, and attackers could, for instance, instigate sudden braking, or acceleration, rendering the lives of passengers and pedestrians at risk [6].

In response, an intrusion detection system (IDS) is usually regarded as the second (and most practical) line of defense, given that an attacker can hack into the vehicle's internal communication. In general, there are two types of vehicular IDSs—signature-based [11], [12] and anomaly-based [13], [14]. A signature-based IDS typically formulates detection rules based on the system's normal behavior and known attacks. Any violations of these rules are regarded as anomalies. In CAN bus, these rules can be based on the frequency of the messages, sequence of message IDs, interframe time differences, signal values, etc. High-dimensional CAN data flow, such as broadcasting different signals/IDs at different frequencies, makes it difficult for the models to extract the effective rules [15]. Moreover, due to the limitations in the rules, these IDSs tend to show a high false-negative rate in detecting advanced attacks and, thus, require frequent updates of the known-attack database as they are only effective against known attack footprints [14]. Moreover, a clever attacker can even keep the sequences of the malicious CAN message benign by turning off the actual ECU through a well-known bus-off attack [16], [17] and sending crafted messages simultaneously on behalf of the victim ECU. Although a few of the works on ECU fingerprinting [18], [19] provided potential ways to verify the source of the CAN message by analyzing the physical-layer attributes of the ECU and detecting such impersonation attacks, the assumption of the uniqueness of such physical properties is proven invalid by a recent study [20]. Moreover, an attacker can also remotely manipulate CAN messages at the data link layer, bypassing the protocol's rules and enabling stealthy link-layer attacks [21]. Some attacks are even possible due to the limitations in the physical layer [22], such as different sample-point settings of ECUs [23]. Therefore, only analyzing the sequence of the CAN messages is not sufficient for the IDS. Rather, the only effective way to detect advanced masquerade attacks, including injection attacks, is to analyze the payload of the messages and check for abnormalities within their contents.

The second category of CAN IDS analyzes anomalies in the CAN data frame. The message IDs and the binary payloads are the main sources of data utilized in such IDSs [24]. Despite the notable advancement in anomaly-based CAN IDS research in recent years, it is still significantly hampered by several factors [25]. First, CAN message in light-duty vehicles are obfuscated by the original equipment manufacturers (OEMs) for security and privacy reasons. Different vehicle models encode their signals using different semantic rules, even under the same OEM. Furthermore, in passenger vehicles, a single payload usually contains several signals, even encoded in different formats, along with some unused bits [26]. Due to this semantic gap, the anomaly-based IDSs built directly on such obfuscated complex binary CAN payloads tend to suffer high false-positive rates and lack of explainability.

Besides, any machine learning (ML)-based IDS running on raw payload data will have challenges if needing to scale with the CAN FD (flexible data-rate) technology where the payload field can be 512 bits long (instead of 64 bits) [27].

On the other hand, the conversion of high-dimensional binary payload data to decimal signals has several benefits [25]. First, it reduces the dimensionality of the data as many bits are combined into a single physically meaningful number. Further, it reduces the inherent noise of the binary bits, which may seem patternless cryptic fluctuations in the raw data but becomes meaningful if appropriately decoded.

Therefore, to achieve a more robust and semantically concise defense against CAN intrusions, it is imperative to design IDS schemes at the signal level, instead of only focusing on the temporal/ID patterns and binary payload. Meanwhile, there are very few concrete proposals for the signal-level CAN IDS [15], [28], [29], [30]. Most of these considered individual deep learning models per CAN ID to track the associated time-series signals, making them impractical for modern vehicles with many CAN IDs. Moreover, as these IDSs have attack-specific designs, they lack a comprehensive detection performance against diverse types of attacks.

Thus, in this article, we propose a deep learning-based intrusion detection framework, CANShield, which can handle high-dimensional vehicular CAN bus data at the signal level and detect advanced and stealthy attacks, including fabrication, suspension, and masquerading attacks with high accuracy and responsiveness. This framework working at the signal level also adds transparency to the detection process.

We make the following contributions to this article.

1) We propose a deep learning-based intrusion detection framework, CANShield, to detect advanced and stealthy attacks from signal-level CAN data. It features a data processing technique (pipeline) for the high-dimensional CAN signal stream by creating a temporary data queue and using the forward-filling mechanism to fill the missing data. This pipeline prepares the data stream suitable for the training and testing in the ML-based IDS.
2) To make the detection effective on multidimensional time-series data of different temporal scales, we convert the 2-D data queues to multiple images and consider the detection as a computer vision-like problem. We consider multiple convolution neural network (CNN)-based

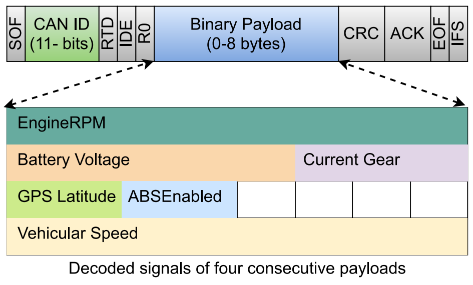

Fig. 1. (Top) CAN data frame syntax. (Bottom) An example of the decoded signals that are encoded in the data field of four consecutive messages.

autoencoder (AE) models to learn the various temporal (short-term and long-term) and spatial (signal-wise) dependencies. Violations in either the temporal or spatial pattern can be detected during the reconstruction process. Such data preprocessing avoids the need for individual ML models per CAN ID.

3) We propose a three-step analysis of the structured reconstruction loss of CANShield's AE models on the selection of detection thresholds for the optimal accuracy, followed by an ensemble-based detector that boosts the overall detection performance by combining the insights from all the AEs. We also utilized transfer learning to reduce the cost of training multiple AE and ensure transferability.
4) We evaluate CANShield against advanced signal-level attacks using SynCAN [15] and ROAD [25] data sets and compare the results with a baseline model to show the improvements. The results show high effectiveness and responsiveness of CANShield against a wide range of fabrication, masquerade, and suspension attacks on the CAN bus. We also make the source code publicly available.[1]

The remainder of this article is organized as follows. We introduce necessary background information in Section II. An overview of the proposed CANShield framework and the attack model is presented in Section III. The technical details are shown in Section IV. We provide an experimental setup and implementation details in Section V. The evaluation results are analyzed in Section VI. The related works are discussed in Section VII. Finally, we conclude this article in Section VIII.

II. Preliminaries

A. Controller Area Network

Robert Bosch GmbH introduced CAN as an automotive communication bus with the latest version (2.0) released in 1991 [31].

CAN Frame Format: A CAN message frame falls into four types: 1) data frame; 2) remote frame; 3) error frame; and 4) overload frame, with data frame being the default mode for data transmission. The top portion of Fig. 1 illustrates the data frame format of CAN. CAN data frame supports up to 8 bytes of payloads with 11 bits of arbitration ID (CAN ID), which can be extended to 29 bits. Every ECU broadcasts its message to the CAN bus. However, only one ECU can transmit at a time and the rest stay synchronized to receive the data correctly. The message arbitration mechanism detects and resolves collisions of messages. A message with a higher priority contains a lower binary-encoded CAN ID. When any ECU detects a higher priority transmission during arbitration, it waits until the end of that message, and the channel is available to use. Due to different priorities, different CAN IDs usually appear in the CAN bus at different frequencies.

Signal-Level Representation of CAN Data: The binary payload can be decoded to the signal level using the specific car's database for CAN (DBC) file [32]. The DBC file is a proprietary format, which is quite challenging to get. However, any reverse engineering-based CAN decoder, such as the CAN-D [26], can provide an approximate DBC file. Such decoding converts the binary payloads to real-valued signals and gives a time-series representation. We define the time of each signal appearance as a one-time step. Thus, there is one CAN message at each time step, which may contain one or more associated signals along with some unused bits. The lower part of Fig. 1 shows some samples of signal-level representation of a few consecutive payloads. To prepare data input to an ML-based detector, a straightforward idea is to create a structured representation of such data stream, where the columns indicate different signals and rows show each time step. As such a data structure contains many missing entries [15], it cannot be directly fed to the ML-based IDS models. Thus, designing an appropriate data preprocessing pipeline to account for the missing signal entries is one of the critical challenges in building a signal-level CAN IDS, as we will address in Section IV-B1.

B. Autoencoder

AE is an artificial neural network that can learn efficient codings of input data through unsupervised learning [33]. It consists of two parts: 1) an encoder that maps an input to a lower dimensional code and 2) a decoder that reconstructs the closest form of the input from that code. In the reconstruction step, encoding parameters are refined so that the decoder can recover the data while retaining only the most relevant features. Hence, a bottleneck in the middle of the network can determine the estimated states of the system in a lower dimension. Let us define the function of encoder and decoder as ϕ and ψ that takes the input $\mathcal{X}$ and $\mathcal{F}$, respectively, such that

$$\phi : \mathcal{X} \rightarrow \mathcal{F}, \quad \psi : \mathcal{F} \rightarrow \mathcal{X}$$
$$\phi, \psi = \arg\min_{\phi,\psi} \|\mathcal{X} - (\psi \circ \phi)\mathcal{X}\|^2.$$

In intrusion detection applications AE plays a vital role. An AE network is first trained on the normal data so that it learns how to reconstruct with minimum loss. The fundamental hypothesis of using AE is that intrusions are sufficiently anomalous with respect to the underlying distribution of the training data so that the AE will yield a high reconstruction loss ($\|\mathcal{X} - (\psi \circ \phi)\mathcal{X}\|$), pointing to a high probability of attack.

[1]https://github.com/shahriar0651/canshield

C. Convolutional Neural Network

Convolutional neural network (CNN) is a class of deep neural networks mostly used to analyze image data sets [34]. The network uses small kernels or filters that slide along the input data and map the complex relationship among the features. CNNs can be considered the regularized versions of multilayer perceptions and takes the advantage of the hierarchical data structure. Small filters help them learn the local and straightforward patterns first and then combine them into more complicated patterns. Therefore, CNN is an extremely powerful tool with a very low degree of connectivity and complexity. We build the AE networks using CNN due to the observation that each view is a 2-D data item, and CNN is widely proven to work efficiently on 2-D data with minimum complexity.

D. Transfer Learning

Transfer learning refers to reusing a model trained for one task as the starting point for another. The pretrained deep learning models are often used as starting points for new models if they are learning similar feature spaces and are working on similar data sets. Therefore, transferring knowledge saves time and cost during the training phase of deep learning [35]. Transfer learning has two basic terms: 1) domain and 2) task. A domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ consists of: a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$. Given a specific domain, $\mathcal{D}$, a task $\mathcal{T} = \{\mathcal{Y}, f(x)\}$ consists of two components: 1) a label space $\mathcal{Y}$ and 2) a predictive function $f : \mathcal{X} \rightarrow \mathcal{Y}$. The function f is used to predict the corresponding label or a representation $f(x)$ of an instance x. This task is learned from the training data consisting of pairs $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in \mathcal{Y}$.

Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$. Out of different ways, one of the most common approaches is to initiate the weights of $f_T(\cdot)$ using the trained parameters of $f_S(\cdot)$. The idea is that the basic structure and knowledge saved in the source model is a good start for the target model; hence, initializing $f_T(\cdot)$ with the parameters of $f_S(\cdot)$ will reduce the initial cost. As in this work, we consider the AE-based models, $f(\cdot)$ will have the function of an AE.

III. System Model

A. CANShield Overview

The main component of CANShield is a software system that can read a vehicle's CAN messages in real-time. It is loaded either on an onboard computing device connected to the OBD-II Port (e.g., laptop and Raspberry Pi) or instantiated in an existing ECU with a relatively powerful processor, such as the gateway ECU. For the former case, the onboard computing device includes a CAN protocol stack, allowing monitoring and recording of the raw CAN messages. This can be achieved with open-sourced implementations, such as Seeed CAN-BUS Shield [36] and SocketCAN [37] or commercial CAN data loggers, such as CANalyzer [38], and VehicleSpy [39], etc. CANShield is preloaded with the vehicle's DBC file, either from OEM or CAN-D, allowing continuous decoding of the binary payloads, creating a data queue of multidimension time-series signals, and tracking their changes in near real-time.

As is shown in Fig. 2, CANShield contains three modules: 1) the *data preprocessing module* that creates multiple data views of the same data queue of signal-level CAN data; 2) the *data analyzing module* that employs multiple CNN-based AEs for analyzing the data views and generating reconstruction losses; and 3) the *attack detection module* that calculates the anomaly scores and makes the final detection decision. CANShield has two phases of operation: 1) *training* and 2) *deployment*. Some of the modules play additional/slightly different roles during each of the two phases. During the training phase, the data analyzing module needs to train deep learning models. However, as the onboard devices are typically lightweight and not suitable for effective training of the deep learning models, we consider two potential solutions for that. CANShield can have a secure connection to the cloud with model training capabilities or train the models on a local computer with CANShield running on that. Hence, during the training phase, the normal CAN traces are stored on the local memory first and then periodically sent to the cloud or local computer for model training. As the AEs have the same tasks (signal reconstruction) but work on slightly different domains (data views), we utilize the transfer learning technique to transfer the knowledge of one AE to the next one which is working on a higher sampling period. Once all the models are adequately trained, CANShield loads the trained models into the onboard device and begins the deployment phase, which goes through the three modules in a feedforward fashion and outputs the detection result in near real-time. It is noted that CANShield detects attacks at the data queue level rather than at the message level.

B. Attack Model

We assume that the intruder can access the CAN bus through an exposed interface, such as V2X, infotainment, ADAS systems, OBD-II port, etc. Moreover, we also assume that the attacker is capable of turning off any ECU [16] and/or injecting arbitrarily malicious messages. CANShield is designed to protect the vehicles from the different levels of attacks in a holistic manner. In particular, according to the attacker's objective, the attacks typically fall into the following three categories.

1) *Fabrication attacks*, wherein a compromised ECU injects malicious IDs and data to the CAN bus. However, all the legitimate ECUs are still active and also send their original data. This is the most prevalent and straightforward type of attack that is quick and easy to launch, as the attacker does not need to hijack any ECU.
2) *Suspension attacks*, wherein a legitimate ECU is turned off/incapacitated by the adversary. This attack is also called *suppress attack*, where the messages from the targeted ECU disappear for a while. To achieve this, the attacker can disconnect the ECU from the in-vehicle network to prevent it from communicating.

3) *Masquerade attacks* are the most advanced, stealthiest, and destructive attacks. This is the combination of fabrication and suspension attacks, where the attacker silences a legitimate ECU, and spoofs it in the continuing operation while injecting malicious messages.

In evaluation, we will use a well-known CAN attack data set, SynCAN [15] and an emergent realistic CAN data set, ROAD [25] covering specific forms of the above attacks to test the efficacy of CANShield.

C. Design Objectives

The design objectives of the CANShield are as follows.

1) *Detecting Advanced Attacks:* The foremost objective of CANShield is to leverage established patterns and correlations of various ECU/signal states during normal driving and design a single IDS that can detect a variety of CAN message injection and manipulation attacks considered in the literature to date, particularly those advanced stealthy attacks that existing ID- or payload-based IDSs have shown ineffective in detecting.
2) *Near Real-Time Detection With Low False Positives (FPs):* The IDS should respond to intrusions accurately, with low false-positive rates, and quickly, at the same order of magnitude as the CAN message intervals, to help the vehicle avoid catastrophes.

IV. CANShield Detailed Design

This section elaborates on CANShield's two initializing tasks and three core modules in detail.

A. Critical Signal Selection and Clustering

As modern vehicles have hundreds of ECUs, they contain a lot of CAN IDs and numerous associated signals. Securing all of them with IDS comes with great implementation and computation costs. On the other hand, securing only a handful of important signals from the critical subsystem of the vehicle, such as the power train, engine, coolant system, etc., will reduce complexity and render feasible solutions for real-time detection. A practical challenge arises in designing an effective detection pipeline with a select group of signals. Accordingly, we consider CANShield to keep track of only m preselected high-priority signals. To find the shortlisted signals, we assume that the defender has the semantic knowledge of the signals, at least on the critical signals to secure. To make the detection more effective and robust CANShield adds additional signals based on the correlation coefficient, starting from the ones with the highest correlation with the critical signals. However, adding too many signals will increase the size of the input image of the AEs, leading to an expensive and ineffective system. Therefore, m is a design parameter and depends on the defender. For the rest of this article, we will use the term "signals" to indicate only the preselected m signals.

The order of the signals in the created 2-D input image could also impact the learning efficacy. Compared to random placement, placements that bring out stronger spatial (correlations) patterns of the signals in the resulting image will enable more effective learning. To facilitate the learning of the intersensor correlations, CANShield calculates the Pearson correlation matrix of the time-series signal data set [40]. Interpreting the correlation coefficient as the distance between a pair of signals, CANShield utilizes a hierarchical agglomerative clustering algorithm with complete linkage method [41] to find compact clusters of highly correlated signals. Later, we use the sequence of clustered signals to build the 2-D images (queue) so that learning the signal-to-signal correlation becomes effective for the small filters of the convolutional layers. Therefore, if one signal starts reporting abnormal values, the CNN model will easily detect anomalies by comparing them with the nearby highly correlated signals. More details on the implementation are in Section VI-A. Notably, the two tasks, signal selection, and correlation-based clustering are done only once during the initialization of the training process (i.e., off-line with recorded data) and are not parts of the detection (deployment) pipeline. The following sections elaborate on the three core modules of CANShield.

B. Data Preprocessing Module

The data preprocessing module prepares formatted 2-D inputs to the AEs of the data analyzing module. It contains the following two steps.

1) Creating and Maintaining Data Queue: First of all, the data preprocessing module continuously records the CAN trace and decodes the binary payloads containing the selected m signals. Then, a first-in–first-out data queue $\mathcal{Q}$ is created with the historical time-series signal data for the last q time steps, where q is large enough for $\mathcal{Q}$ to encompass the temporal pattern of different signals. Thus, every new CAN message is a new entry in $\mathcal{Q}$, where the signal values only associated with that incoming CAN ID are updated. For the rest signals, which are not updated by the new message, we adopt a forward-filling technique, whereas, at every time step, the missing/unreported signals are copied from the previous time step. We assume that until an ECU sends a further CAN message, its signals are still the same as the latest reported ones. Thus, as time passes, the sensor data for the last q time steps are always stored in $\mathcal{Q}$.

2) Creating Multiple Views: To learn the various temporal (short-term and long-term) patterns of different signals with different reporting periods and identify abnormality, the data analyzing module needs to train and deploy the AE networks on different views (short-term and long-term) of the data queue $\mathcal{Q}$. As different CAN IDs have different reporting periods, only the first w ($<<q$) messages or time steps (columns) of $\mathcal{Q}$ may not be enough to represent the recognizable temporal trend for all the signals, especially for the ones with longer reporting cycles. On the other hand, considering a high value for w ($\approx q$) makes the input image too large. As a result, the AE models become more complex. This challenge boils down to how to effectively learn the temporal pattern of all the signals, especially of the ones with long reporting periods, while still using a small time window during image generation.

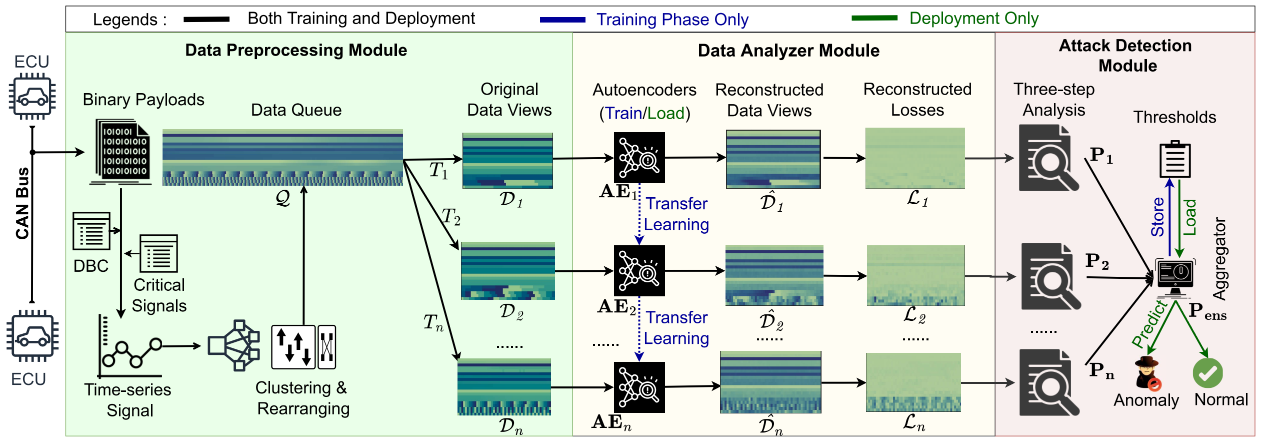

Fig. 2. CANShield workflow. CANShield has two phases of operation: "training" and "deployment." CANShield contains three modules: i) the data preprocessing module that creates multiple data views of the same data queue of signal-level CAN data, ii) the data analyzing module that employs multiple CNN-based AEs for analyzing the data views and generating reconstruction losses, and iii) the attack detection module that calculates the anomaly scores and makes the final detection decision.

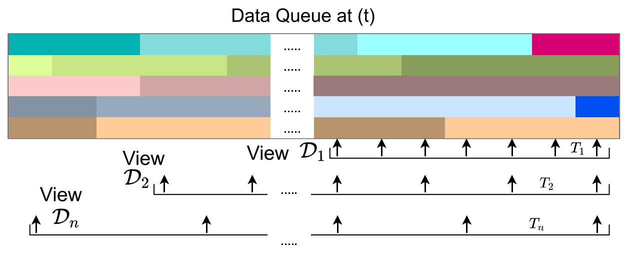

Fig. 3. Generation of different views of $\mathcal{Q}$ with multiple samplings at time step t. For the visualization, we have transposed the original image, where the signals associated with each CAN ID are presented as a single row, and the columns indicate the time steps. The changes in the colors indicate the updates in the signal values associated with the CAN IDs. Thus, we select the first w columns from $\mathcal{Q}$ at every $T_1, T_2, \ldots, T_n$ time steps, respectively. Here, $T_1, T_2, \ldots, T_n$ are the sampling periods to create the views $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n$, respectively, of the same $\mathcal{Q}$. Without the loss of generality, here we assume $T_1 < T_2 < \cdots < T_n$. Therefore, $\mathcal{D}_1$ has a more detailed view but contains a very limited historical trend, capturing short-term, or fast-changing patterns. On the other hand, $\mathcal{D}_n$ has most of the temporal trend, capturing long-term or slow-changing patterns, but with the lowest details.

We achieve these two conflicting goals by creating n different views of $\mathcal{Q}$ with n different sampling periods (seeing more with a less complex model). Fig. 3 illustrates such sampling process at time step t that uses sampling periods $T_1, T_2, \ldots, T_n$ to create the views $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n$, respectively, of the same $\mathcal{Q}$. The forward-filling mechanism helps to preserve the short-term or fast-changing attributes in this long-term view. Despite having different sampling periods, CANShield keeps the number of samples (w) within each data view the same. As there are total m signals, each data view will have a dimension of $m \times w$. This allows CANShield to use the same architecture for all the AE models working on each data view. The multiview design has benefits in the system's accuracy and scalability. On the other hand, each of these views has different primary targeted signals, but collectively they cover temporal trends of variable lengths. This allows more effective and accurate detection of abnormal signals, regardless of attacking message frequency and duration.

C. Data Analyzing Module

The data analyzing module utilizes multiple AE models: $\{\mathbf{AE}_i\}_{i \in [n]}$ (we define $[n] \overset{\text{def}}{=} \{1, 2, \ldots, n\}$). Each of the models is associated with each of the views of $\mathcal{Q}$ and thus learns different (and complementary) perspectives of $\mathcal{Q}$. We build the AE networks using CNN due to the observation that each view is a 2-D data item, and CNN is widely proven to work efficiently on 2-D data with minimum complexity. The motivation for using AE is that, as there are neither explicitly defined states of the vehicle, nor any analytical model for that, we use a data-driven approach to find the states out of a small window of the historical signal data. Thus, the data in an AE's central (bottleneck) layer represents the vehicle's state in a lower dimension. In contrast, the decoder part tries to predict the vehicle's historical signal data by looking at the state's information. If the vehicle is running in a normal state, as mostly seen in the training data, the decoder should predict accurately. Otherwise, an abnormal state will lead to an erroneous prediction, therefore, a high reconstruction loss. Moreover, as our considered model learns the relationship among all the signals, especially the nearby highly correlated ones if at least one signal deviates from the regular pattern, CANShield will recognize it from the reconstruction loss.

As shown in Fig. 2, during the training phase, each $\mathbf{AE}_x$ takes a data view $\mathcal{D}_x \in \mathbb{R}^{m \times w}$ as an input image and learns to reconstruct almost the same $\hat{\mathcal{D}}_x \in \mathbb{R}^{m \times w}$ image $\forall x \in [n]$. Meanwhile, as CANShield trains different AEs for different views, the training cost would be linear to the number of views. Thus, a practical challenge lies in *how to reduce the cost of training multiple AEs*. As the views are created from the same data queue $\mathcal{Q}$, they contain inherent similarities in their structure.

First, the spatial dependencies (correlations) along the features are still almost the same, as all the signals in each of the views are sampled with the same sampling periods. On the other hand, the temporal patterns in different views are just the expanded/shrunk versions. Hence, instead of training all the models from scratch, we consider training the first model

AE$_1$ thoroughly. Then, we use the transfer learning technique to initialize the parameters of the next model, **AE**$_2$, which only needs to fine-tune the parameters instead of learning everything from scratch. Thus, we initialize any tth model **AE**$_t$ with the preceding trained model **AE**$_{t-1}$. Such a technique reduces the training cost (see Section VI-D), which will be most effective if, in the future, the model is trained in a peripheral device like Raspberry Pi for a new vehicle.

Once the training is done, the deployment phase is initiated, and the trained models are loaded in CANShield. At the end of the training phase and during the deployment phase, the AEs are tested on the corresponding data stream and try to reconstruct the same image. For **AE**$_x$, the absolute difference between the original image and the reconstructed image is the reconstruction loss $\mathcal{L}_x \in \mathbb{R}^{m\times w}$ is calculated as follows:

$$\mathcal{L}_x = abs\left(\mathcal{D}_x - \hat{\mathcal{D}}_x\right). \tag{1}$$

Each element contains the corresponding signal's reconstruction loss at a certain time step, where the row and columns indicate the signal and time steps, respectively.

D. Thresholds Selection and Attack Detection Module

In this part, we discuss how to interpret a 2-D reconstruction loss $\mathcal{L}_x$ into an anomaly score P_x (i.e., attack probability) for every data view $\mathcal{D}_x$ and use the results for attack detection.

For a normal computer vision problem, the common practice would be to consider the mean value of all the elements of the absolute reconstruction loss matrix $\mathcal{L}$ as the anomaly score P

$$P \leftarrow \frac{1}{m\times w}\sum_{i=1}^{m}\sum_{j=1}^{w}\mathcal{L}_{i,j}. \tag{2}$$

Compared to a normal computer vision problem, our input image (and reconstruction loss $\mathcal{L}_x$) has a concrete structure, which gives space for tweaking the detection thresholds for better accuracy. Thus, instead of taking the average value, we exploit the structural knowledge of $\mathcal{L}_x$ to interpret the P_x. We define three types of thresholds for attack detection at each $\mathbf{AE_x}$.

1) Signal-wise reconstruction loss thresholds $R_x^{\text{Loss}} \in \mathbb{R}^m$.
2) Signal-wise time step violation thresholds $R_x^{\text{Time}} \in \mathbb{R}^m$.
3) An overall signal violation threshold $R_x^{\text{Signal}} \in \mathbb{R}$.

Next, we demonstrate a *three-step analysis* on $\mathcal{L}_x$ to facilitate the selection of these thresholds and attack detection, as is shown in Algorithms 1 and 2, respectively. For convenience, we have obviated the AE index x for the thresholds and $\mathcal{L}$ as this approach will be applied independently to each AE. We also use three system hyperparameters p, q, r as confidence percentiles for these thresholds, which is subject to optimal tuning in practice (see Section VI-B1).

First, Algorithm 1 shows how we select the thresholds from the 3-D reconstruction loss matrix $\mathcal{L}$ from randomly selected t training data queues. First, we find the R_i^{Loss} for every signal $i \in [m]$ on the normal training data by taking the pth percentile values of elements in the ith rows of all the $\mathcal{L}$ (3). Later, we map the 3-D matrix $\mathcal{L}$ to a binary 3-D matrix $\mathcal{B}$ to find the indices where the reconstruction losses are higher than the allowed threshold R_i^{Loss} for every ith signal (4). Second, we find the total number of such time step violations $\mathcal{V}_i$ for each signal by summing over all the w time steps (5) for all the t instances. We evaluate the distribution of the signal-wise total time step violations and consider the qth-percentile value as the time step violation threshold R_i^{Time} (6).

Algorithm 1: Thresholds Selection for **AE**$_x$

```
Input: Stack of reconstruction losses $\mathcal{L} \in \mathbb{R}^{t\times m\times w}$, system hyperparameters
        $p, q, r$
Variables: $\mathcal{B} \leftarrow 0^{t\times m\times w}$,  $\mathcal{V}, \mathcal{S} \leftarrow 0^{t\times m}$,
Output: Thresholds: $R^{\text{Loss}}, R^{\text{Time}} \in \mathbb{R}^m, R^{\text{Signal}} \in \mathbb{R}$
/* Step 1                                                         */
    $\forall i \in [m] : R_i^{\text{Loss}} \leftarrow p^{th}\ \%\ \forall j,k \in [w],[t]\ \mathcal{L}_{i,j}^k$                    3
    $\forall i,j,k \in [m],[w],[t] : \mathcal{B}_{i,j}^k \leftarrow 1$ if $\mathcal{L}_{i,j}^k > R_i^{\text{Loss}}$                    4
/* Step 2                                                         */
    $\forall i,k \in [m],[t] : \mathcal{V}_i^k \leftarrow \sum_{j=1}^{w} \mathcal{B}_{i,j}^k$                    5
    $\forall i \in [m] : R_i^{\text{Time}} \leftarrow q^{th}\ \%\ \forall k \in [t]\ \mathcal{V}_i^k$                    6
/* Step 3                                                         */
    $\forall i,k \in [m],[t] : \mathcal{S}_i^k \leftarrow 1$ if $\mathcal{V}_i^k > R_i^{\text{Time}}$                    7
    $\forall k \in [t] : P^k \leftarrow \frac{1}{m}\sum_{i=1}^{m} \mathcal{S}_i^k$                    8
    $R^{\text{Signal}} \leftarrow r^{th}\ \%\ \forall k \in [t]\ \ P^k$                    9
```

As the third step, we check if any specific signal has more time step violations than R_i^{Time} and flag that signals as compromised (7) in each instance. Now, we have the list of the violating signals $\mathcal{S}$ in each data view, and we consider the average value of $\mathcal{S}$ as the anomaly score P for the **AE** (8). Considering the false-positive requirement of the system, we set rth percentile value of all Ps of the considered samples, as the total signal violation threshold R^{Signal} (9). After running all the steps, CANShield stores R_x^{Loss}, R_x^{Time}, and R_x^{Signal} for each of the $\mathbf{AE_x}$, and consider the average of all R_x^{Signal}s as the threshold $R_{\text{ens}}^{\text{Signal}}$ for the ensemble model.

During the deployment phase, these thresholds are preloaded from the memory and Algorithm 2 is used to detect any violation. Although the tasks in (10)–(13) are similar as Algorithm 1, CANShield runs them on individual test reconstruction loss $\mathcal{L}$ and check for potential threats using the ensemble model. Here, an anomaly score is assigned on each of the reconstruction losses on the data views, i.e., P_1, P_2, $\cdots$, P_n. CANShield then uses the ensemble anomaly score P_{ens} (14) as the final score. In the case of $P_{\text{ens}} > R_{\text{ens}}^{\text{Signal}}$, the IDS tags $\mathcal{Q}$ as anomalous and raises the alarm in the system (15). Compared to the mean absolute value method (2), this three-step method gives CANShield finer decomposition of $\mathcal{L}$ and improves the detection efficacy against stealthy attacks. Fig. 4 shows a simplified visualization of Algorithm 2 with a 5×5 reconstruction loss matrix.

Algorithm 2: Ensemble-Based Detection

```
Input: Reconstruction loss $\mathcal{L} \in \mathbb{R}^{m \times w}$,
Thresholds: $R^{\text{Loss}}, R^{\text{Time}} \in \mathbb{R}^m, R^{\text{Signal}} \in \mathbb{R}$
Variables: $\mathcal{B} \leftarrow 0^{m \times w}, \ \mathcal{V}, \mathcal{S} \leftarrow 0^m$
Output: Attack decision: $attack \in \mathbb{R}$
/* Step 1                                                    */
    $\forall i, j \in [m], [w] : \mathcal{B}_{i,j} \leftarrow 1$ if $\mathcal{L}_{i,j} > R_i^{\text{Loss}}$    10
/* Step 2                                                    */
    $\forall i \in [m] : \mathcal{V}_i \leftarrow \sum_{j=1}^{w} \mathcal{B}_{i,j}$    11
    $\forall i \in [m] : \mathcal{S}_i \leftarrow 1$ if $\mathcal{V}_i > R_i^{\text{Time}}$    12
  /* Step 3                                                  */
    $P_x \leftarrow \frac{1}{m} \sum_{i=1}^{m} \mathcal{S}_i$    13
  /* Ensemble                                                */
    $P_{\text{ens}} \leftarrow \frac{1}{n} \sum_{x=1}^{n} P_x$    14
    $attack \leftarrow 1$ if $P_{\text{ens}} > R_{\text{ens}}^{\text{Signal}}$    15
```

V. Implementation and Evaluation

A. Data Sets and Attacks

We implement CANShield on both the SynCAN data set and ROAD data set. SynCAN data set [15] (Synthetic CAN Bus Data) is a widely used CAN attack data set released by ETAS (a subsidiary of Robert Bosch Gmbh) covering stealthy signal-level CAN attacks. ROAD data set [25] was released by Oak Ridge National Laboratory and is the most realistic CAN attack data set to date.[2] Next, we introduce the details of each data set and the attacks covered.

1) SynCAN: The SynCAN data set is built on actual CAN traces, emulating the characteristics of the real CAN traffic, with hundreds of advanced attack scenarios. It contains a total of 20 signals, including physical values, counters, and flags. There are 24 h of logged data, of which 16.5 h are for training and 7.5 h are for testing with five types of advanced attacks, which resembles the three stealthy forms of attack models mentioned in Sections III-B.

The attacks in SynCAN data sets are summarized in Table I. In a *flooding attack*, the attacker frequently broadcasts high-priority messages to delay the legitimate ECUs' transmission (similar as DoS attack). In a *suppress attack*, the attacker turns off the corresponding ECU of the targeted signal(s) or prevents it from sending further messages. Based on the time-series nature of the injected data, there are three types of masquerade attacks. In a *plateau attack*, the attacker broadcasts the same constant value of any signal over a long period of time. The impact of such an attack depends on the extent of the leap and the duration of the attack. In a *continuous attack*, the signals are overwritten with continuously changing values that shift

[2]To the best of our knowledge, the SynCAN data set (available at https://github.com/etas/SynCAN) was the only publicly available signal-level CAN data set with advanced attacks at the time of writing this article. ROAD data set (available at https://0xsam.com/road/) was obfuscated and did not have signal-level interpretation in its initial release in early 2021. We obtained the raw ROAD data set by directly contacting ORNL. Partially motivated by our work, ORNL has recently released a signal-level ROAD data set.

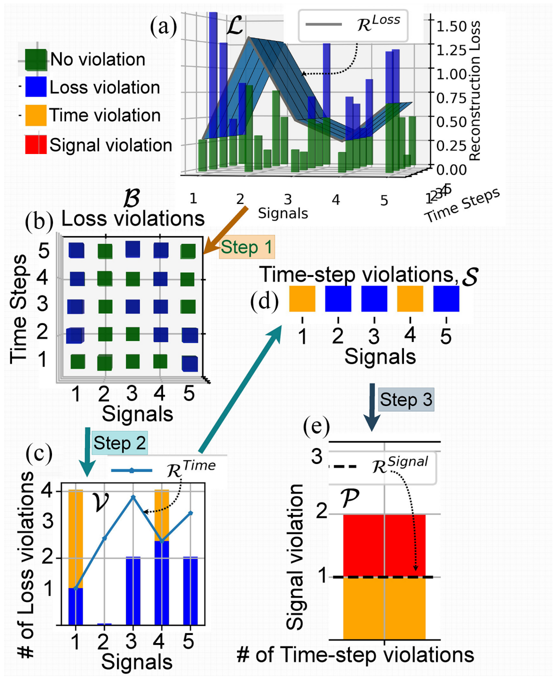

Fig. 4. Simplified visual illustration of three-step attack detection (Algorithm 2) for individual AE with 5×5 reconstruction loss matrix. (a) 3-D visualization of 2-D reconstruction loss matrix $\mathcal{L}$ showing the loss violations ($\mathcal{L} > R^{\text{Loss}}$) in blue. (b) Binary 2-D matrix $\mathcal{B}$ showing the indices of loss violation [top view of (a)]. (c) Signal-wise total loss violations $\mathcal{V}$ [counting only the blue bars in (b)]. Orange colors show where $\mathcal{V}$ violates time-step threshold R^{Time}. (d) Binary 1-D array $\mathcal{S}$ showing if any signal violates R^{Time} [top view of (c)]. (e) Anomaly score/total signal violations $\mathcal{P}$ showing the total number of time-step violating signals [counting only the orange bars in (d)]. The red color shows if $\mathcal{P}$ exceeds the threshold R^{Signal}, indicating a potential attack; otherwise, the final prediction will be benign. For simplification, we show the total counts in the bar plots instead of using the percentage, which is used in the actual algorithm.

TABLE I
Description of Attacks in SynCAN Data Set

Attack Name	Attack Type	Description
Flooding	Fabrication	Frequently injects high-priority messages.
Suppress	Suspension	Prevent an ECU from transmission.
Plateau	Masquerade	Broadcasts a constant value.
Continuous		Broadcasts continuously changing values.
Playback		Broadcasts a series of recorded values.

from the actual ones. Such small changes can initially look realistic and bypass IDS. Finally, in a *playback attack*, the attacker replays a series of previously recorded data for the targeted signal to make it more realistic.

2) ROAD: The ROAD data set provides the highest-fidelity CAN traces with physically verified most realistic CAN attacks. It contains a significant amount of training data covering the different contexts of driving. We obtained the raw ROAD data set and extracted signals from the CAN messages using CAN-D. There are 3.5 h of logged data, of which 3 h are for training and 30 min are for testing with five types of advanced masquerade attacks targeting the engine coolant temperature, engine RPM, brake light, and wheel speed sensors. The injected message manipulates only the specific portion of the data fields containing the targeted signals.

TABLE II
DESCRIPTION OF MASQUERADE ATTACKS IN ROAD DATA SET

Attack Name	Description and impacts
Correlated signals	Inject different values for wheel speeds that stop the car.
Max speedometer	Inject maximum value to display on the speedometer.
Max coolant temp	Inject maximum value; turn the coolant warning light on.
Reverse light on/off	Toggle the reverse light that does not reflect the gear.

Whereas the attacks in the SynCAN are created by post-processing (replacing original ones) on the normal driving data, the attack traces in the ROAD traces were collected from a real vehicle under the real injection attacks. Such attack traces provide not only the injected messages but also the response from the vehicle under such attacks, which makes the ROAD data set the most realistic one. The attacks in the ROAD data set are summarized in Table II. In light of the model's complexity, one single IDS is not a feasible option to track all the hundreds of decoded signals within the ROAD data set. Thus, in the implementation of CANShield on the ROAD data set, we consider seven primary signals, which were compromised during the attacks, to be of primary importance and add two highly correlated signals for each to make the IDS more robust, as detailed in Section IV-A.

B. Evaluation Setup

1) CANShield Software Implementation: We use Python 3.7.3 with Keras 2.2.4 [42] for training and evaluation of CANShield. The pipeline for the AE model contains the combinations of the convolutional layer, activation layer (LeakyRelu), max pooling, and up-sampling layers [34]. Using min–max scaling, we keep the values of each signal between 0 and 1. We used a five-layer network, where the convolutional layers have 3×3 filters, and the numbers filters in each layer are 32, 16, 16, 32, and 1. We utilized leakyRelu as the activation function with a parameter of 0.2, except for the output layer, which has a sigmoid activation function. The decoder part contains up-sampling layers with 2×2 filters. We use the Adam optimizer with a learning rate of 0.0002 to train the models and mean square error as the loss function. Using a batch size of 128, we train each model for 100 epochs. The following section explains the impact of different parameters in attack detection and illustrates the effectiveness of CANShield.

2) Evaluation Settings: To evaluate CANShield, we consider w as 25, 50, and 100, and five sampling periods (T_x) as 1, 5, 10, 20, and 50 for each of the data sets. After training the AE models, we select a random 10% of the samples from the training data and determine the reconstruction losses using (1) and time step violations for each AE. We also study the comparative analysis of the effectiveness of different sampling periods against different attacks. We do an extensive grid search with all the combinations of threshold ranging from 90% to 99.99% as p, q, and r to find R^{Loss}, and R^{Time}, and R^{Signal}, respectively, as mentioned in (3), (6), and (9), to evaluate CANShield and maximize detection performance. Moreover, we evaluate different detection scenarios by setting 0.1%, 0.5%, and 1% as the maximum threshold for the FP rate (FPR) in the system.

With these settings, we evaluate CANShield's performance in the following three aspects.

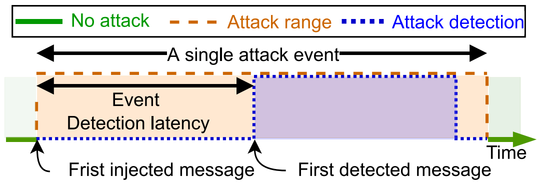

Fig. 5. Attack detection and event detection latency in a single attack event.

Attack Detection: Any injection or modification of any CAN message, as is described in the attack model in Section III-B, is considered an attack. Attack detection is defined as the detection of any malicious data view. If any view of the data queue contains one or more malicious injections, we consider the label of the queue view as malicious.

Event Detection Latency: Depending on the type of attack, there could be a delay between the first injected message and the first correct detection during any attack event. Such a delay is defined as the event detection latency. Fig. 5 shows the event detection latency for a single attack event.

Hardware Processing Latency: We evaluate CANShield's performance by implementing it on a standard computer as well as a lightweight edge device and benchmark the inference time, showing the near real-time performance in hardware.

3) Evaluation Metrics: For any binary classifier, there are four possible outcomes. True positive (TP), and true negative (TN) are the outcomes where the model correctly predicts the positive (attack) and negative (benign) classes, respectively. An FP and false negative (FN) are the outcomes where the model incorrectly predicts the positive classes and negative classes, respectively. Based on these outcomes, we evaluate CANShield's performance using the following metrics.

1) *Precision* is defined as the ratio between the correctly predicted positive data views to a total number of predicted positive views (TP/[TP + FP]).
2) *Recall* or *TP Rate (TPR)* is calculated as the ratio between the number of positive views correctly classified as positive to the total number of actual positive views (TP/[TP + FN]).
3) *FPR* is the proportion of negative views incorrectly identified as positives (FP/[FP + TN]).
4) *F1 Score* is the harmonic mean of precision and recall ($2 \times ([\text{Precision} \times \text{Recall}]/[\text{Precision} + \text{Recall}])$. For an imbalanced data set, F1 score is mostly used to evaluate the model's performance.
5) *ROC Curve, PR Curve, and AUC Scores* indicate the classifiers performance with varying discrimination thresholds [43]. The ROC curve plots TPRs and FPRs, and the PR curve plot precisions and recalls for different thresholds. The area under the ROC and PR curves are represented as AUROC and AUPRC, respectively, which indicate the robustness of the detectors. An ideal detector has both AUROC and AUPRC scores of 1.00.

4) Baseline Models: We consider CANShield with only one AE with sampling periods T_x as CANShield-T_x and the full-fledged multi-AE-based CANShield as CANShield-Ensemble (or CANShield-Ens).

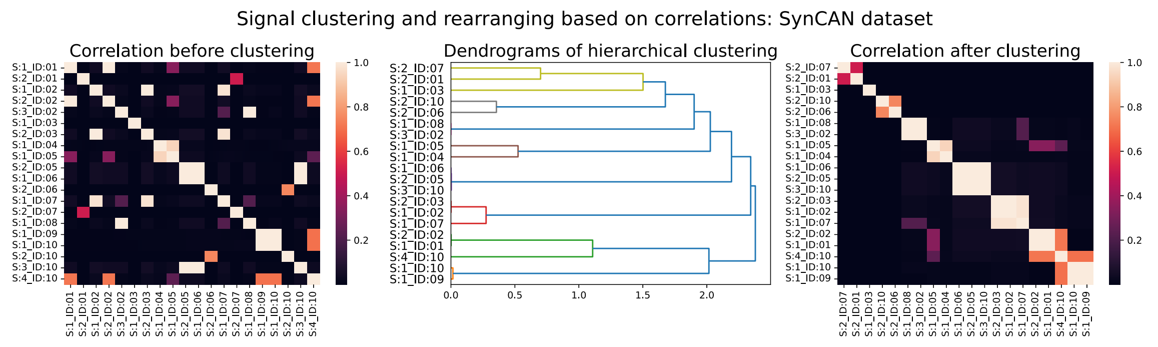

Fig. 6. Hierarchical clustering of the signals in SynCAN data set based on the correlation matrix and rearranging them in clusters.

This part describes the four baseline models that we consider for the performance comparison.

1) *CANShield-Base:* We consider CANShield-Base, a simplified version of CANShield to represent the existing approaches in CNN-AE-based IDS working on windows of multidimensional time-series data [44]. We consider CANShield-Base to have only one AE working with a sampling period of 1 using the conventional one-step mean absolute value of reconstruction loss (as (2)) to calculate the anomaly score. Hence, the performance comparison between CANShield-Ens and CANShield-Base justifies the significance of multiple AEs and a three-step analysis of reconstruction losses.
2) *CANet:* CANet [15] is the IDS specifically designed for high-dimensional CAN data structure, employing one long short-term memory (LSTM) model for each CAN IDS and merging their output to create a fully connected AE network. The authors evaluated CANet on the SynCAN data set and made the data set public [45]. As we are also utilizing the SynCAN data set, CANet becomes the most relevant baseline for CANShield-Ens.
3) *Reconstructive:* The fundamental approach of the reconstructive baseline is similar to CANShield-Base. Whereas CANShield-Base feeds all the signals in one single AE model, reconstructive baseline uses different AE models for different signals [46]. Therefore, although it can learn the temporal dynamics, there is no way to learn the signal-wise correlations.
4) *Predictive:* In the predictive baseline, there are individual LSTM models for each CAN ID that predicts the signals associated with the CAN ID for the next time-step [47]. Hence, whereas all the reconstruction-based methods, including CANShield and CANet, rely on the reconstruction of the input that contains the past and current values, the predictive baseline forecasts the future values from the given input and compares them with the reported ones.

VI. Evaluation Results and Discussion

This section, first, explains why correlation-based clustering is effective for CANShield; and later shows CANShield's performance on the different aspects.

A. Correlation-Based Clustering

As discussed in Section IV-A, in the initialization of the training phase, CANShield analyzes the Pearson correlations matrix of the data set to create clusters of signals and rearrange them so that highly correlated signals stay together in the data queue $\mathcal{Q}$. The left panel of Fig. 6 shows the heat map of the correlation matrix of the SynCAN data set, with the original orders of the signals as appeared in the data set. It is clear from the figure that some of the highly correlated signal pairs, for example, S:1_ID:02, and S:1_ID:07, have a correlation of around unity but originally, they are placed far apart. Such placement makes it harder for the small CNN filters to learn their dependencies. The middle panel of Fig. 6 shows the dendrograms after correlation-based clustering, which also indicates the existence of multiple clusters of highly correlated signals. For example, in the SynCAN data set, S:1_ID:10 and S:1_ID:09 form a cluster of two signals, and S:2_ID:03, S:1_ID:07, and S:1_ID:02 form another cluster. The right panel of the figure shows the heat map of the correlation matrix after the signal reordering. Therefore, such grouping and reordering make data queue $\mathcal{Q}$ generation more interpretable and effective.

B. Attack Detection

1) Optimizing Design Hyperparameters: We first show how we optimize CANShield's system hyperparameters to achieve the best performance on the SynCAN data set. We assess the contribution of each feature of CANShield in attack detection in the three following steps.

Effectiveness of Three-Step Analysis: As the first version of CANShield, we consider CANShield-1, which uses only one AE working on a sampling period of 1 and a data view length of 50. Thus, the three-step analysis of reconstruction loss is the only difference between CANShield-1 and CANShield-Base. Hence, we demonstrate the efficacy of the three-step analysis of reconstruction loss (in CANShield-1) over the mean absolute loss (in CANShield-Base) by selecting different values for thresholds R^{Loss}, R^{Time}, and R^{Signal}, respectively. The captions in Fig. 7 show the AUROC score of CANShield-Base for each attack type, while different pixels indicate the improvements in the AUROC scores of CANShield-1 over CANShield-Base for different combinations of R^{Loss} and R^{Time}.

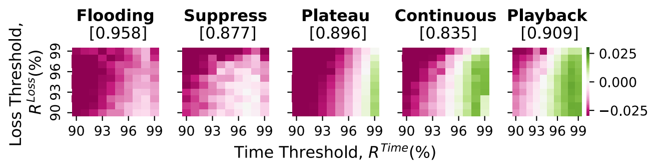

Fig. 7. Effectiveness of three-step loss analysis in CANShield over the mean absolute loss in CANShield-Base. The values within the [ ] show the AUROC scores of CANShield-Base, whereas the colors of the pixels show the improvements in the AUROC scores for different R^{Loss} and R^{Time}.

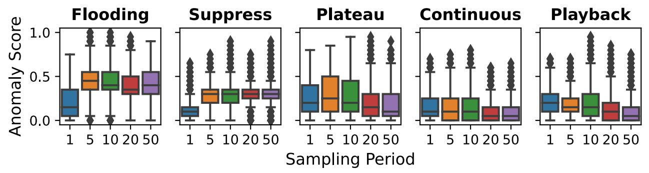

Fig. 8. Anomaly scores of CANShield with different sampling periods on malicious samples. Higher anomaly scores on malicious samples make the IDS more effective.

The figure shows whereas the proposed three-step analysis has limited contributions on the *flooding* and *suppress* attacks (first two panels), it provides a better representation of violations and improves the detection performance of the stealthy *masquerade* attacks (last three panels) compared to CANShield-Base. As the violations in the *fabrication* and *suspension* attacks are more evident and do not involve any modification of signals, mean absolute loss itself suffices to give a decent detection performance (AUROC scores of 0.958, and 0.877, respectively). However, setting R^{Loss} and R^{Time} to 95-percentile and 99-percentile, respectively, helps better analyze the nuanced violations created by the *masquerade* attacks and provides the improvements (0.02–0.03 in AUROC scores) over CANShield-Base. This evaluation shows adding a three-step analysis improves the detection rate even when one AE is used. In the following paragraphs, we will discuss how adding more AEs, and combining them into an *ensemble* detector, CANShield-Ens further improves the detection performance.

Effectiveness of Different Sampling Periods: Here, we demonstrate the effectiveness of learning from multiple views with multiple AEs working with different sampling periods in detecting attacks. Fig. 8 illustrates the performance comparison of CANShield-T_x, where $T_x \in \{1, 5, 10, 20, 50\}$. We analyze the effectiveness of CANShield-T_x by plotting the distributions of anomaly scores of the malicious data queues. As the anomaly scores on the benign data queues are mostly zeros, we only show the anomaly scores on malicious data queues. The first two panels of the figure show that for both *flooding* and *suppress* attacks, the anomaly scores of the malicious data queues increase for higher sampling periods, making the detection easier as these attacks are more detectable looking at the long-term sequential pattern. As higher anomaly scores on malicious data queues make the classification task easier, it increases the TPR while lowering the FPR. Therefore, AE working on a higher sampling period (≥ 5) is the most effective against *fabrication* and *suspension* attacks.

On the other hand, a sampling period of 5 seems to be the most suitable choice against *plateau* attacks, and a sampling

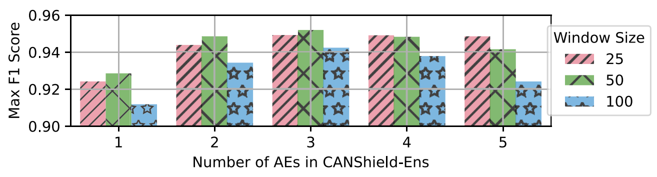

Fig. 9. Optimizing CANShield-Ens's architecture. Best AUROC score for different window size w ({25, 50, 100}) and AEs.

period of 1 is the best performing one against the *continuous*, and *playback* attacks. Hence, unlike *fabrication* and *suspension* attacks, the lower sampling periods (≤ 5) are, in general, the most effective ones against the *masquerades* attacks as short-term views of the data queue provide a detailed look at the time-series abnormalities. Therefore, only one AE working on only one type of data representation is not enough to detect diverse attacks. This finding motivates the design of CANShield-Ens, combining multiple AEs into a single decision model to further increase the robustness of the IDS.

Effectiveness of Ensemble Model: To design the final ensemble model, we studied different combinations of AEs working with different sizes of data views. Here, we consider the standard ensemble technique of averaging multiple anomaly scores to a single score (attack probability) as mentioned in (14) and use that to evaluate the detection performance.

To search for the final ensemble model, we studied different window sizes and different combinations of AEs, starting from one AEs up to five AEs. As Fig. 9 shows, CANShield with only one AE has limited performance (AUROC score < 0.93) regardless of the window size w. When more AEs are ensembled, the performance improves. Although $w = 25$ shows promising performance, it still underperforms that of $w = 50$ even having more AEs. Besides, we observe that $w = 100$ tends to make the model overly complicated and yield performance degradation. From the figure, it is evident that, on average, CANShield-Ens performs the best on the SynCAN data set when $w = 50$ and there are three AEs working. We further find that out of various combinations of three sampling periods, the ensemble of 1, 5, and 10 gives the best performance.

We note that although the above results are derived from the SynCAN data set, the ROAD data set also shows a similar result. Therefore, for the simplicity of the analysis, we use w as 50, three AEs (with sampling periods 1, 5, and 10), and R_{Loss} and R_{Time} as 95th-percentile and 99th-percentile, respectively, for both the SynCAN and ROAD data sets in the following evaluations.

2) Attack Visualization and AUROC Scores: In this part, we visualize the anomaly scores for all the individual and ensemble detectors along with the ROC curves for both SynCAN and ROAD data sets.

SynCAN Data Set: Fig. 10(a) shows the CANShield's anomaly scores, and the left panel of Table III summarizes AUROC scores for different attacks on the SynCAN data set. Different AEs (CANShield-T_x) show different performances on each of the attacks. However, the CANShield-Ens yields more stable and consistent performance, leading to

TABLE III
PERFORMANCE COMPARISON WITH DIFFERENT CANSHIELD ARCHITECTURES AND BASELINE DETECTORS ON SYNCAN DATA SET

	Area Under the ROC Curve (AUROC)						True Positive Rate / False Positive Rate				
	Flooding	Suppress	Plateau	Continuous	Playback	Average	Flooding	Suppress	Plateau	Continuous	Playback
CANShield-1	0.940	0.860	0.907	0.853	0.927	0.898	0.581 / 0.009	0.166 / 0.008	0.42 / 0.007	0.361 / 0.007	0.657 / 0.006
CANShield-5	0.997	0.976	0.944	0.837	0.905	0.932	0.992 / 0.01	0.82 / 0.008	0.482 / 0.009	0.353 / 0.007	0.594 / 0.004
CANShield-10	0.994	0.978	0.924	0.814	0.888	0.920	0.976 / 0.008	0.846 / 0.009	0.429 / 0.009	0.247 / 0.006	0.589 / 0.005
CANShield-Ens	**0.997**	**0.985**	0.961	0.870	0.948	**0.952**	0.988 / 0.009	0.781 / 0.01	0.486 / 0.009	0.427 / 0.01	0.689 / 0.008
CANShield-Base	0.958	0.877	0.896	0.835	0.909	0.895	0.656 / 0.01	0.338 / 0.01	0.534 / 0.01	0.463 / 0.01	0.698 / 0.01
CANet [15]	0.979	0.882	**0.983**	**0.936**	**0.974**	0.951	0.901 / 0.004	0.613 / 0.004	0.955 / 0.025	0.765 / 0.006	0.905 / 0.004
Reconstructive [46]	0.903	0.496	0.755	0.563	0.532	0.650	0.688 / 0.005	0.001 / 0.007	0.361 / 0.074	0.016 / 0.025	0.029 / 0.005
Predictive [47]	0.874	0.489	0.722	0.561	0.530	0.635	0.644 / 0.006	0.003 / 0.007	0.33 / 0.026	0.015 / 0.006	0.02 / 0.004

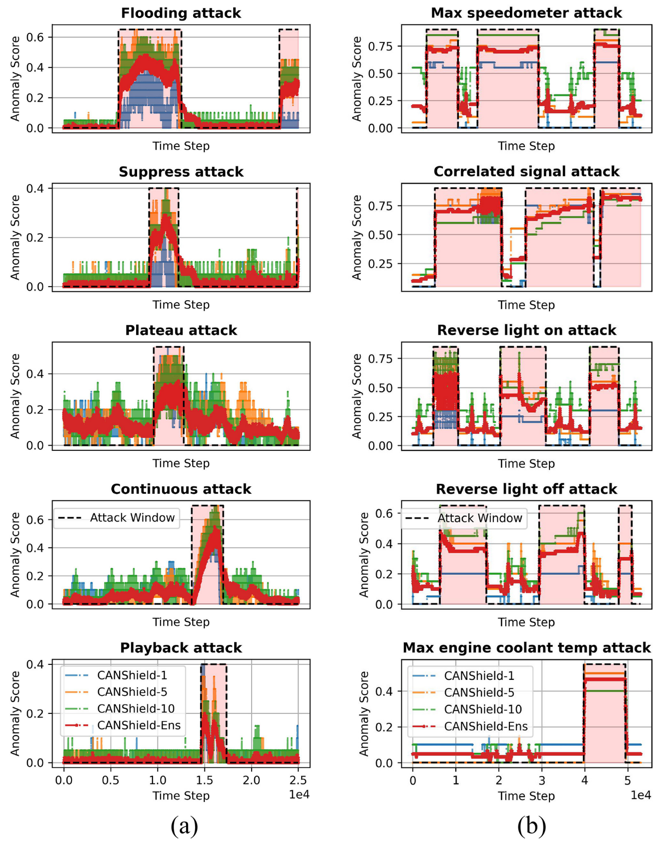

Fig. 10. Attack visualization with different models for both data sets. (a) SynCAN data set. (b) ROAD data set.

higher AUROC scores in all the attacks than the individual CANShield-T_x. In the case of *continuous* and *playback* attacks, the signals start to deviate gradually from the original values, which takes some time to create the recognizable deviation for the IDS. Hence, a lower AUROC score in the CANShield-Ens is not unexpected, especially against *continuous* attacks. However, CANShield-Ens can detect the violations almost instantly for the rest of the attacks (AUROC scores of $0.95-1.00$). Whereas the individual AEs are attack-specific, the ensemble model takes the best out of every model, generalizes the process, and detects most attacks with the highest AUROC scores.

ROAD Data Set: Fig. 10(b) shows the anomaly scores of the attacks on the ROAD data set. Same as the SynCAN, CANShield-Ens also shows stable performance in the anomaly score. As all the attacks in the ROAD data set are closely aligned with the *plateau* attack in the SynCAN data set, both the individual and ensemble models show high performance in detecting the attacks. There are a few cases where the performance degrades a little bit, but CANShield-Ens mitigates such issues and detects all the attacks on the ROAD data set with an AUROC score of $\sim$1.00.

3) Precision, Recall, and F1 Score: In this part, we study the impact of the signal violation thresholds R^{Signal} on CANShield-Ens's precision, recall, and F1 score for different attacks in both the SynCAN and ROAD data set. The first panel of Fig. 11(a), which shows the PR curve along with the AUPRC scores on the SynCAN data set, demonstrates that CANShield-Ens is highly effective against *fabrication* and *suspension* attacks (AUPRC $\geq$0.92) and moderate performance against advanced *masquerade* attacks (AUPRC$\approx$ $0.65-0.88$). Moreover, the values of R^{Signal} within the range of 0.05 to 0.2 provide a decent performance maximizing the F1 scores for different attacks, as shown in the right panel of the figure. Considering CANShield's goal of having a low FPR, we recommend a higher value for R^{Signal}, which results in high precision ($>$0.9) for all the attacks. Similarly, the evaluation results in Fig. 11(b) on the ROAD data set show that CANShield-Ens achieves perfect precision, recall, and F1 score (AUPRC$\approx$1.00) with an appropriate threshold $(0.2-0.3)$.

Comparison With Baseline Models: Whereas we demonstrate the improvements of CANShield-Ens over the individual models, this part includes the performance comparison with the other baseline detectors as well. Table III illustrates such comparison, which indicates 1.84% and 11.67% improvements in the AUROC of *flooding* and *suppress* attacks, respectively, compared to the closest baseline CANet. Unlike CANet, CANShield-Ens considers both the sequence of CAN IDs and the time-series signal values to create the data queue and provides effective detection of such practical attacks. Even though CANet performs slightly better against advanced masquerade attacks, CANShield-Ens also shows decent performance. The right panel of Table III shows the TPR and FPR of different CANShield architectures along with the baselines. Similar to the AUROC, CANShield shows promising performance against fabrication and suspension attacks, while CANet performs better against masquerade attacks. Furthermore, CANShield is considerably lighter than CANet. While CANet consumes 8718 kB of memory [28], CANShield only utilizes 525 kB, making it suitable for edge devices. Overall, as Table III shows, CANShield-Ens outperforms all of the baselines on average, showing the proposed framework's effectiveness.

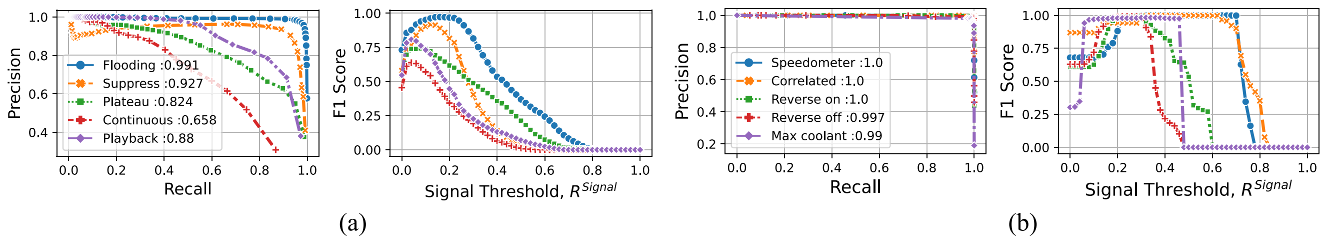

Fig. 11. CANShield-Ens's precision-recall (PR) curve with AUPRC and F1 Scores for different thresholds on both the (a) SynCAN and (b) ROAD data sets.

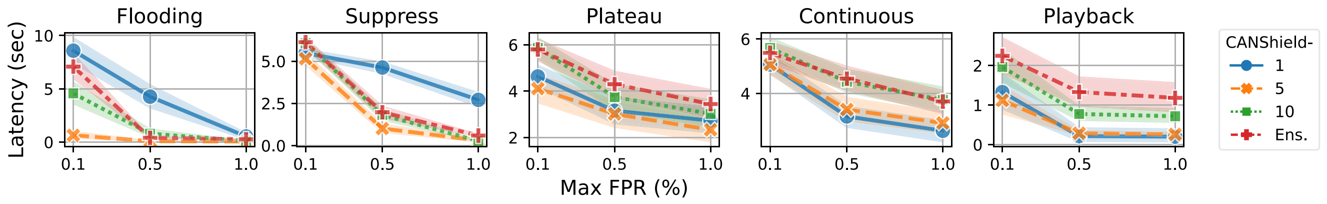

Fig. 12. Tradeoff between event detection latencies and maximum FPR thresholds against different attacks in the SynCAN data set.

C. Event Detection Latency

Fig. 12 illustrates the attack-wise event detection latency for three cases of maximally allowed FPR for the SynCAN data set. As each attack manipulates the signal at different paces, the time to observe a potential deviation varies. Hence, similar to the previous discussion, certain AEs are more responsive against certain types of attacks. As the first two panels of Fig. 12 show that in the case of *fabrication* and *suspension* attacks, CANShield-1 has slightly higher event detection latency, whereas CANShield-Ens reduces the detection latency for the ensemble model. On the other hand, the *masquerade* attacks are the most challenging to detect, and CANShield-Ens reduces the FPs by taking the mean of the final anomaly scores. Therefore, as a tradeoff, it increases the latency by a small factor compared to the individual models. However, the latency is still within a small range to cause any devastating impact. It is noted again that in the SynCAN data set, the attacks were created in post-processing without any physical verification. Hence, some attacks may align with the actual data and lose the malicious property leading to low-detection performance and high-detection latency.

Furthermore, the figures also illustrate the impact of maximum FPR on the event detection latency. Although some individual model suffers from high latency with low FPR (i.e., 0.1%), CANShield-Ens provides a lower event detection latency. However, allowing more FPs (max FPR of 0.5%−1%) into the system further reduces latency. Whereas in case some advanced SynCAN attacks CANShield takes up to a couple of seconds to detect, all the attack events in the ROAD data set are detected almost instantly [see Fig. 10(b)]. Therefore, our evaluation shows that CANShield improves detection performance, reduces overall detection latency, and makes the system more robust.

D. Implementation and Processing Latency

Transfer Learning: Here, we explain the computational benefit of transferring knowledge from the trained AE models working on lower T_x to AEs with higher T_x. Fig. 13(a) shows

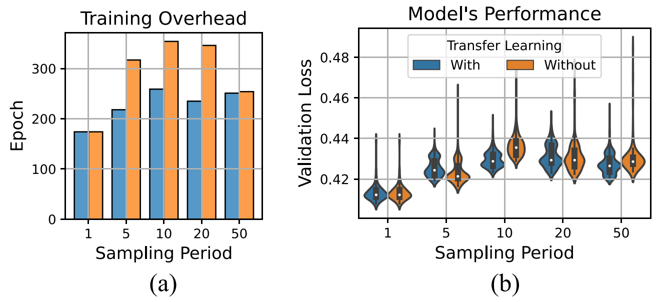

Fig. 13. Effectiveness of transfer learning during model training. (a) Cost of training. (b) Validation loss after training.

that without any knowledge transfer, the number of training epochs to reach the early stopping criteria, which is a steady validation loss, increases by up to 100% of the initial training for different AEs. However, if the AE model's parameters are initialized as the pretrained AE with the immediate lower T_x, the number of training epochs gets reduced by approximately 30% in most cases. Besides, as Fig. 13(b) shows, such initialization does not impact the performance of the final models as the validation loss of the final AE models remains almost the same regardless of the weight initialization. Therefore, CANShield-Ens reduces the training cost of consecutive AEs significantly by transferring the weights to the next AE without any performance tradeoff.

Hardware Processing Latency: We trained and evaluated CANShield on a laptop with a 2.3 GHz 8-Core Intel i9 processor with 32 GB of RAM and AMD Radeon Pro 5500M 8 GB of graphics and also deployed on a Raspberry Pi with 1.5 GHz 64-bit quad-core CPU and 4 GB of RAM to benchmark CANShield prediction speed. To reduce the inference time and the size of AE models, we convert the TensorFlow model into TensorFlow Lite [48] models, which quantizes the weights. Results show each CANShield process takes around 1 ms on the laptop, which satisfies our design objective (<2 ms), and 10 ms on the Raspberry Pi, which is low for an attack to cause catastrophe to the targeted vehicle. Our extensive testing and validation demonstrate that the quantized AE-based CANShield provides no degradation in performance and yields the same detection results as the original ones.

E. Limitations and Discussions

Here, we discuss two key challenges of CANShield, which are common for any DL-based signal-level CAN IDS.

1) The first challenge is to get the DBC files from the OEM or have an efficient reverse engineering tool to create the signal-level representation of the CAN data set. Hence, we assume that the defender is OEM who has direct access to the DBC file or a third party with an efficient reverse engineering tool.
2) The collection of sufficient training data and generalizing the training of the AE models is another challenge. To overcome these issues, CANShield is assumed to be trained on a very dynamic high-fidelity data set, including a diverse range of driving patterns and various driving scenarios, to ensure that it can detect anomalies regardless of the driving context and driver's behavior.

VII. Related Work

There has been a good amount of work on CAN IDS, which can be divided into the following general categories.

Physical Characteristics-Based IDS: One line of research in CAN IDS utilized the physical layer attributes of the CAN bus communications to fingerprint the ECUs and verify the source of each message. Since the physical signals generated from the ECUs solely depend on the ECUs' hardware characteristics, it is assumed to be unique; hence, a malicious ECU cannot controllably modify it. Therefore, such defense has been considered effective in detecting injection attacks. Out of different attributes, clock skews [19], voltage profile [49], [50], electrical CAN signal characteristics [18], [51], etc., are widely used in fingerprinting and building physical characteristics-based IDS. However, the assumption of the uniqueness of such physical properties is proven invalid by a recent study [20], which proposed a voltage corruption tactic that can modify the physical attributes of the victim ECU and impersonate the targeted ECU. Therefore, such IDSs cannot provide a comprehensive security guarantee against a wide range of cyberattacks.

CAN ID-Based IDS: A vast portion of the attacks, especially *fabrication* and *suspension* attacks, consider exploiting the sequences of CAN IDs to disrupt regular services. Therefore, some IDSs extract features from the series of CAN IDs to learn the usual pattern and detect abnormalities. Given the labeled data sets, some works utilized different types of supervised learning models, based on CNN [52], [53], LSTM [54], support vector machine, *k*-nearest neighbors, decision tree, random forest, and XGBoost [55], [56], [57] etc., to build the IDSs. Different unsupervised ML algorithms are also studied in CAN ID-based IDS research. Various features, such as message timing information per CAN ID and window-wise ID-counting, are used as the underlying features for the IDSs [58].

A few works predicted the next CAN ID with individual LSTM or gated recurrent unit (GRU) models and used log loss and a predefined threshold to detect malicious injections [59]. Similarly, one-class support vector machine (OCSVM) [60], isolation forest [61] are also studied. Along with unsupervised methods, self-supervised method-based IDS are also studied [62]. A few works converted the sequences of CAN IDs into 2-D images and trained generative adversarial networks (GANs) in an unsupervised fashion [63], [64]. Recently, motivated by natural language processing, some researchers considered the sequence of CAN IDs as a sentence and utilized world embedding and language models to build the CAN IDS [65], [66]. The fundamental drawback of the CAN ID-based IDSs is that they are only effective against injection attacks that explicitly change the sequence of IDs. However, advanced masquerade attacks can manipulate the payload without disrupting the ID sequences/frequencies and easily evade such IDSs [6].

Payload-Based Detection: The advanced attacks can not only change the CAN IDs but also modify the payloads of the messages. The attacker can replay prerecorded values or change the actual values. Hence, there has been a good amount of work learning the pattern in the payload sequences and using it to detect potential cyberattacks. Extracting usable features from the binary payloads is a challenging task. The mode and value information is commonly used to extract features and implement DNN-based IDS [67]. A few works proposed a continuous field classification (CFC) algorithm to identify the payload value alignments and used a deep learning-based approach to identify the anomalous fields [68]. Moreover, different *k*-nearest neighbor classifiers are also used to identify different attacks [69]. Considering the sequence of CAN messages as time-series data, a few works implemented unsupervised ML models based on LSTM [70], [71] and OCSVM to build the payload-based CAN IDS [72].

Signal-Level Detection: Compared to the IDSs mentioned above, IDSs working at the time-series signal level can extract the most useful information and build an efficient and context-aware decision model. Moriano et al. [30] hypothesized that masquerade attacks alter the correlations among the signals and the clustering behaviors and proposed a technique to detect such attacks by comparing the clustering similarity of test data with and without attack traces. Recent works proposed DNN-based signal-level CAN IDS, where the extracted sensor values are used as separate features for IDS [73]. Other research efforts also proposed the RNN/LSTM-based models with an embedding layer working on CAN payload values in [47], [74], and [75]. A few similar approaches in CAN IDS research used GRU, LSTM, and temporal CNN-based AEs for each CAN ID [28], [29], [74], [75], [76], [77]. All of these IDSs [28], [29], [74], [75], [76], [77] processed ID-wise data independently and utilized individual models for each ID, which ignored the signal-wise correlations and fail to detect attack collectively.

CANet [15] is one of the closest works to our proposed method. It employed one LSTM model for the signals with each CAN ID and used AE-based reconstruction to predict the anomaly score. However, in practice, LSTM networks are costly to train, and one LSTM for each IDS will make it impractical for a vehicle with many CAN IDs. Moreover, due to the complicated architecture, CANet shows low-detection performance on *suppress* attacks, a form of the well-known bus-off attack that can be easily launched due to the CAN protocol's limitations. Novikova et al. [78] proposed to manually group the highly correlated signals into smaller subgroups and use AE for each subgroup. However, such manual clustering is not feasible for real vehicles with lots of signals.

VIII. Conclusion

As modern vehicles become more connected to external networks, the attack surface of the CAN bus system grows drastically. To secure the CAN bus from advanced intrusion attacks, we propose a signal-level intrusion detection framework, CANShield. With the capability of handling a high-dimensional CAN data stream, CANShield trains multiple CNN-based AE models to work on different views of the data stream across different temporal scales, performs a three-step structural analysis of the reconstruction losses, and finally ensembles them to obtain the final anomaly score. Evaluation results on both the SynCAN and ROAD data sets show CANShield's robustness and responsiveness against different advanced attacks. The proposed three-step analysis of the reconstruction loss improves the overall AUROC by 6.40% than the conventional mean average method. The aggregation of data with different temporal scales reduces variance in inference and increases the AUROC by at least 2.19% compared to any single AE-based framework. Moreover, CANShield outperforms all the baselines against practical fabrication and suspension attacks while still performing well against advanced masquerade attacks. By combining the strengths of CAN ID-based IDS and signal-level IDS, CANShield offers a scalable and efficient solution and advances the state-of-the-art.

Acknowledgment

The authors are thankful to Robert A. Bridges from ORNL for his insightful comments on this article. This manuscript has been coauthored by UT-Battelle, LLC, under Contract DE-AC05-00OR22725 with the U.S. Department of Energy (DOE). The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for U.S. Government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

References

[1] Z. El-Rewini, K. Sadatsharan, D. F. Selvaraj, S. J. Plathottam, and P. Ranganathan, "Cybersecurity challenges in vehicular communications," *Veh. Commun.*, vol. 23, Jun. 2020, Art. no. 100214.

[2] C. E. Andrade et al., "Managing massive firmware-over-the-air updates for connected cars in cellular networks," in *Proc. 2nd ACM Int. Workshop Smart, Auton., Connected Veh. Syst. Services*, 2017, pp. 65–72.

[3] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, S. Garg, and M. M. Hassan, "BDTwin: An integrated framework for enhancing security and privacy in cybertwin-driven automotive Industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17110–17119, Sep. 2022.

[4] K. Koscher et al., "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Security Privacy*, 2010, pp. 447–462.

[5] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle CAN," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 993–1006, Apr. 2015.

[6] C. Miller, "Lessons learned from hacking a car," *IEEE Design Test*, vol. 36, no. 6, pp. 7–9, Dec. 2019.

[7] G. David. "Chrysler recalls 1.4 million hackable cars." Accessed: Jun. 21, 2021. [Online]. Available: http://cnnmon.ie/1OrrqGv

[8] H. J. Jo and W. Choi, "A survey of attacks on controller area networks and corresponding countermeasures," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6123–6141, Jul. 2022.

[9] Y. Xiao, S. Shi, N. Zhang, W. Lou, and Y. T. Hou, "Session key distribution made practical for can and CAN-FD message authentication," in *Proc. Annu. Comput. Security Appl. Conf.*, 2020, pp. 681–693.

[10] J. Schmandt, A. T. Sherman, and N. Banerjee, "Mini-MAC: Raising the bar for vehicular security with a lightweight message authentication protocol," *Veh. Commun.*, vol. 9, pp. 188–196, Jul. 2017.

[11] S. Jin, J.-G. Chung, and Y. Xu, "Signature-based intrusion detection system (IDS) for in-vehicle CAN bus network," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2021, pp. 1–5.

[12] H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, "SAIDuCANT: Specification-based automotive intrusion detection using controller area network (CAN) timing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1484–1494, Feb. 2020.

[13] S. Halder, M. Conti, and S. K. Das, "COIDS: A clock offset based intrusion detection system for controller area networks," in *Proc. 21st Int. Conf. Distrib. Comput. Netw.*, 2020, pp. 1–10.

[14] W. Wu et al., "A survey of intrusion detection for in-vehicle networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 919–933, Mar. 2020.

[15] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "CANET: An unsupervised intrusion detection system for high dimensional CAN bus data," *IEEE Access*, vol. 8, pp. 58194–58205, 2020.

[16] K.-T. Cho and K. G. Shin, "Error handling of in-vehicle networks makes them vulnerable," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 1044–1055.

[17] G. Bloom, "WeepingCAN: A stealthy CAN bus-off attack," in *Proc. Workshop Autom. Auton. Veh. Security*, 2021, pp. 1–6.

[18] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park, and D. H. Lee, "Identifying ECUs using inimitable characteristics of signals in controller area networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 4757–4770, Jun. 2018.

[19] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *Proc. 25th USENIX Security Symp.*, 2016, pp. 911–927.

[20] R. Bhatia, V. Kumar, K. Serag, Z. B. Celik, M. Payer, and D. Xu, "Evading voltage-based intrusion detection on automotive CAN," in *Proc. NDSS*, 2021, pp. 1–17.

[21] A. de Faveri Tron, S. Longari, M. Carminati, M. Polino, and S. Zanero, "CANFlict: Exploiting peripheral conflicts for data-link layer attacks on automotive networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2022, pp. 711–723.

[22] A. Z. Mohammed, Y. Man, R. Gerdes, M. Li, and Z. B. Celik, "Physical layer data manipulation attacks on the CAN bus," in *Proc. Int. Workshop Autom. Auton. Veh. Security (AutoSec)*, 2022, pp. 1–5.

[23] L. Yue, Z. Li, T. Yin, and C. Zhang, "CANCloak: Deceiving two ECUs with one frame," in *Proc. Workshop Autom. Auton. Veh. Security (AutoSec)*, 2021, pp. 1–6.

[24] 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 J. Wireless Commun. Netw.*, vol. 2019, no. 1, pp. 1–17, 2019.

[25] M. E. Verma et al., "Addressing the lack of comparability & testing in can intrusion detection research: A comprehensive guide to CAN IDS data & introduction of the road dataset, 2022, *arXiv:2012.14600*.

[26] M. E. Verma, R. A. Bridges, J. J. Sosnowski, S. C. Hollifield, and M. D. Iannacone, "CAN-D: A modular four-step pipeline for comprehensively decoding controller area network data," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 9685–9700, Oct. 2021.

[27] J. W. Shin, J. H. Oh, S. M. Lee, and S. E. Lee, "Can FD controller for in-vehicle system," in *Proc. Int. SoC Design Conf. (ISOCC)*, 2016, pp. 227–228.

[28] V. K. Kukkala, S. V. Thiruloga, and S. Pasricha, "INDRA: Intrusion detection using recurrent autoencoders in automotive embedded systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3698–3710, Nov. 2020.

[29] V. K. Kukkala, S. V. Thiruloga, and S. Pasricha, "LATTE: LSTM self-attention based anomaly detection in embedded automotive platforms," *ACM Trans. Embedded Comput. Syst.*, vol. 20, no. 5S, pp. 1–23, 2021.

[30] P. Moriano, R. A. Bridges, and M. D. Iannacone, "Detecting can masquerade attacks with signal clustering similarity," in *Proc. Workshop Autom. Auton. Veh. Security (AutoSec)*, 2022, pp. 1–8.

[31] M. Di Natale, H. Zeng, P. Giusto, and G. Arkadeb, *Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice*. New York, NY, USA: Springer, 2012.

[32] M. D. Pesé, T. Stacer, C. A. Campos, E. Newberry, D. Chen, and K. G. Shin, "LibreCAN: Automated can message translator," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2019, pp. 2283–2300.

[33] D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, "A survey of deep learning methods for cyber security," *Information*, vol. 10, no. 4, p. 122, 2019.

[34] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *Proc. Int. Conf. Eng. Technol. (ICET)*, 2017, pp. 1–6.

[35] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, Hershey, PA, USA: IGI Global, 2010, pp. 242–264.

[36] "Seeed Arduino CAN." 2021. [Online]. Available: https://github.com/Seeed-Studio

[37] "SocketCAN." Accessed: Dec. 10, 2021. [Online]. Available: https://python-can.readthedocs.io/en/master/interfaces/socketcan.html

[38] "CANalyzer." 2021. [Online]. Available: https://www.vector.com/int/en/products/products-a-z/software/canalyzer/

[39] "VehicleSpy." 2021. [Online]. Available: https://intrepidcs.com/products/software/vehicle-spy/

[40] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise Reduction in Speech Processing*. Heidelberg, Germany: Springer, 2009, pp. 37–40. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00296-0_5

[41] J. H. Ward Jr., "Hierarchical grouping to optimize an objective function," *J. Amer. Stat. Assoc.*, vol. 58, no. 301, pp. 236–244, 1963.

[42] "Keras team." Keras. 2015. [Online]. Available: https://github.com/keras-team/keras

[43] J. A. Hanley and B. J. McNeil. "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 43, no. 1, pp. 29–36, 1982.

[44] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," in *Proc. Wireless Telecommun. Symposium (WTS)*, 2018, pp. 1–5.

[45] "SynCAN." Accessed: Jun. 21, 2021. [Online]. Available: https://github.com/etas/SynCAN

[46] M. Weber, G. Wolf, E. Sax, and B. Zimmer, "Online detection of anomalies in vehicle signals using replicator neural networks," in *Proc. 6th Escar USA*, 2018, pp. 1–14.

[47] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)*, 2016, pp. 130–139.

[48] "Tensorflow lite." 2022. [Online]. Available: https://www.tensorflow.org/lite

[49] K.-T. Cho and K. G. Shin, "Viden: Attacker identification on in-vehicle networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 1109–1123.

[50] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "VoltageIDS: Low-level communication characteristics for automotive intrusion detection system," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 2114–2129, 2018.

[51] M. Kneib and C. Huth, "Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 787–800.

[52] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Veh. Commun.*, vol. 21, Jan. 2020, Art. no. 100198.

[53] A. Desta, S. Ohira, I. Arai, and K. Fujikawa, "REC-CNN: In-vehicle networks intrusion detection using convolutional neural networks trained on recurrence plots," *Veh. Commun.*, vol. 35, Jun. 2022, Art. no. 100470.

[54] M. Jedh, L. B. Othmane, N. Ahmed, and B. Bhargava, "Detection of message injection attacks onto the CAN bus using similarities of successive messages-sequence graphs," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4133–4146, 2021.

[55] R. U. D. Refat, A. A. Elkhail, A. Hafeez, and H. Malik, "Detecting CAN bus intrusion by applying machine learning method to graph based features," in *Proc. SAI Intell. Syst. Conf.*, 2021, pp. 730–748.

[56] M. L. Han, B. I. Kwak, and H. K. Kim, "Event-triggered interval-based anomaly detection and attack identification methods for an in-vehicle network," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2941–2956, 2021.

[57] I. Aliyu, M. C. Feliciano, S. Van Engelenburg, D. O. Kim, and C. G. Lim, "A blockchain-based federated forest for SDN-enabled in-vehicle network intrusion detection system," *IEEE Access*, vol. 9, pp. 102593–102608, 2021.

[58] D. H. Blevins, P. Moriano, R. A. Bridges, M. E. Verma, M. D. Iannacone, and S. C. Hollifield, "Time-based CAN intrusion detection benchmark," in *Proc. Workshop Autom. Auton. Veh. Security (AutoSec)*, 2021, pp. 1–7.

[59] S. Rajapaksha, H. Kalutarage, M. O. Al-Kadri, G. Madzudzo, and A. V. Petrovski. "Keep the moving vehicle secure: Context-aware intrusion detection system for in-vehicle CAN bus security," in *Proc. 14th Int. Conf. Cyber Conflict Keep Moving! (CyCon)*, vol. 700, 2022, pp. 309–330.

[60] M. Al-Saud, A. M. Eltamaly, M. A. Mohamed, and A. Kavousi-Fard, "An intelligent data-driven model to secure intravehicle communications based on machine learning," *IEEE Trans. Ind. Electron.*, vol. 67, no. 6, pp. 5112–5119, Jun. 2020.

[61] S. Sharmin and H. Mansor, "Intrusion detection on the in-vehicle network using machine learning," in *Proc. 3rd Int. Cyber Resilience Conf. (CRC)*, 2021, pp. 1–6.

[62] H. M. Song and H. K. Kim, "Self-supervised anomaly detection for in-vehicle network using noised pseudo normal data," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1098–1108, Feb. 2021.

[63] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based intrusion detection system for in-vehicle network," in *Proc. 16th Annu. Conf. Privacy, Security Trust (PST)*, 2018, pp. 1–6.

[64] Q. Zhao, M. Chen, Z. Gu, S. Luan, H. Zeng, and S. Chakrabory, "CAN bus intrusion detection based on auxiliary classifier GAN and out-of-distribution detection," *ACM Trans. Embedded Comput. Syst.*, vol. 21, no. 4, pp. 1–30, 2022.

[65] D. Shi, M. Xu, T. Wu, and L. Kou, "Intrusion detecting system based on temporal convolutional network for in-vehicle CAN networks," *Mobile Inf. Syst.*, vol. 2021, Sep. 2021, Art. no. 1440259.

[66] G. Baldini, "Intrusion detection systems in in-vehicle networks based on bag-of-words," in *Proc. 5th Cyber Security Netw. Conf. (CSNet)*, 2021, pp. 41–48.

[67] M.-J. Kang and J.-W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," *PLoS ONE*, vol. 11, no. 6, 2016, Art. no. e0155781.

[68] F. Fenzl, R. Rieke, Y. Chevalier, A. Dominik, and I. Kotenko, "Continuous fields: Enhanced in-vehicle anomaly detection using machine learning models," *Simulat. Model. Pract. Theory*, vol. 105, Dec. 2020, Art. no. 102143.

[69] F. Martinelli, F. Mercaldo, V. Nardone, and A. Santone, "Car hacking identification through fuzzy logic algorithms," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, 2017, pp. 1–7.

[70] Y. Wang, D. W. M. Chia, and Y. Ha, "Vulnerability of deep learning model based anomaly detection in vehicle network," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, 2020, pp. 293–296.

[71] Z. Khan, M. Chowdhury, M. Islam, C.-Y. Huang, and M. Rahman, "In-vehicle false information attack detection and mitigation framework using machine learning and software defined networking," 2019, *arXiv:1906.10203*.

[72] V. Chockalingam, I. Larson, D. Lin, and S. Nofzinger, "Detecting attacks on the CAN protocol with machine learning," *Annu. EECS*, vol. 558, no. 7, pp. 1–8, 2016.

[73] J. Zhang, F. Li, H. Zhang, R. Li, and Y. Li, "Intrusion detection system using deep learning for in-vehicle security," *Ad Hoc Netw.*, vol. 95, Dec. 2019, Art. no. 101974.

[74] V. Tanksale, "Anomaly detection for controller area networks using long short-term memory," *IEEE Open J. Intell. Transp. Syst.*, vol. 1, pp. 253–265, 2020.

[75] J. Ashraf, A. D. Bakhshi, N. Moustafa, H. Khurshid, A. Javed, and A. Beheshti, "Novel deep learning-enabled LSTM autoencoder architecture for discovering anomalous events from intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4507–4518, Jul. 2021.

[76] S. Longari, D. H. N. Valcarcel, M. Zago, M. Carminati, and S. Zanero, "CANnolo: An anomaly detection system based on LSTM autoencoders for controller area network," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1913–1924, Jun. 2021.

[77] S. V. Thiruloga, V. K. Kukkala, and S. Pasricha, "TENET: Temporal CNN with attention for anomaly detection in automotive cyber-physical systems," in *Proc. 27th Asia–South Pacific Design Autom. Conf. (ASP-DAC)*, 2022, pp. 326–331.

[78] E. Novikova, V. Le, M. Yutin, M. Weber, and C. Anderson, "Autoencoder anomaly detection on large CAN bus data," in *Proc. DLP-KDD*, 2020, pp. 1–9.

Md Hasan Shahriar (Student Member, IEEE) received the B.Sc. degree in electrical and electronic engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2016, and the M.S. degree in computer engineering from Florida International University, Miami, FL, USA, in 2020. He is currently pursuing the Ph.D. degree in computer science with Virginia Tech, Arlington, VA, USA, under the supervision of Prof. W. Lou.

His research interests include automotive cybersecurity, cyber–physical systems, and machine learning.

Yang Xiao (Member, IEEE) received the B.S. degree from the School of Electrical and Information Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2014, the M.S. degree from the Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI, USA, in 2017, and the Ph.D. degree from the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Arlington, VA, USA, in 2022, supervised by Prof. W. Lou.

He is an Assistant Professor with the Department of Computer Science, University of Kentucky, Lexington, KY, USA. His research interests lie in network security, blockchain and distributed system security, and wireless network security.

Pablo Moriano (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Pontificia Universidad Javeriana, Bogotá, Colombian, 2008 and 2011, respectively, and the M.S. and Ph.D. degrees in informatics from Indiana University Bloomington, Bloomington, IN, USA, in 2017 and 2019, respectively.

He is a Research Scientist with the Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA. His research lies at the intersection of data science, network science, and cybersecurity. In particular, he uses data-driven and computational methods to discover and understand critical security issues in large-scale networked systems. Applications of his research range across multiple disciplines, including, the detection of exceptional events in social media, Internet route hijacking, and insider threat behavior in version control systems.

Dr. Moriano is a member of ACM and SIAM.

Wenjing Lou (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2003.

She is currently a W. C. English Endowed Professor of Computer Science with Virginia Tech, Arlington, VA, USA. Her research interests cover many topics in the cybersecurity field, with her current research interest focusing on wireless network security, trustworthy AI, blockchain, and security and privacy problems in the Internet of Things (IoT) systems.

Prof. Lou is a highly cited researcher by the Web of Science Group. She received the Virginia Tech Alumni Award for Research Excellence in 2018. She received the INFOCOM Test-of-Time paper award in 2020. She was the TPC chair for IEEE INFOCOM 2019 and ACM WiSec 2020. She was the Steering Committee Chair for IEEE CNS conference from 2013 to 2020. She is currently a steering committee member of IEEE INFOCOM and IEEE TRANSACTIONS ON MOBILE COMPUTING. She served as a program director at the US National Science Foundation (NSF) from 2014 to 2017.

Y. Thomas Hou (Fellow, IEEE) received the Ph.D. degree from the NYU Tandon School of Engineering, Brooklyn, NY, USA, in 1998.

He is currently a Bradley Distinguished Professor of Electrical and Computer Engineering with Virginia Tech, Blacksburg, VA, USA, which he joined in 2002. He was a member of Research Staff with the Fujitsu Laboratories of America, Sunnyvale, CA, USA, from 1997 to 2002. He has published over 350 papers in IEEE/ACM journals and conferences. He holds six U.S. patents. He authored/coauthored two graduate textbooks: *Applied Optimization Methods for Wireless Networks* (Cambridge University Press, 2014) and *Cognitive Radio Communications and Networks: Principles and Practices* (Academic Press/Elsevier, 2009). His current research focuses on developing innovative real-time solutions to complex science and engineering problems arising from wireless and mobile networks. He is also interested in wireless security.

Prof. Hou was/is on the editorial boards of a number of IEEE and ACM transactions and journals. His papers were recognized by ten best paper awards from IEEE and ACM, including an IEEE INFOCOM Test of Time Paper Award in 2023. He was a Steering Committee Chair of IEEE INFOCOM conference and a member of the IEEE Communications Society Board of Governors. He was also a Distinguished Lecturer of the IEEE Communications Society. He was named an IEEE Fellow for contributions to modeling and optimization of wireless networks.