IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 29 July 2022, accepted 18 August 2022, date of publication 20 September 2022, date of current version 30 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3208091

RESEARCH ARTICLE

A Binarized Neural Network Approach to Accelerate In-Vehicle Network Intrusion Detection

LINXI ZHANG[1], (Member, IEEE), XUKE YAN[2], (Member, IEEE), AND DI MA[1], (Senior Member, IEEE)

[1]Computer and Information Science Department, University of Michigan–Dearborn, Dearborn, MI 48128, USA
[2]Department of Computer Science and Engineering, Oakland University, Rochester, MI 48309, USA

Corresponding author: Di Ma (dmadma@umich.edu)

ABSTRACT Controller Area Network (CAN) is the de facto standard for in-vehicle networks. However, it is inherently vulnerable to various attacks due to the lack of security features. Intrusion detection systems (IDSs) are considered effective approaches to protect in-vehicle networks. IDSs based on advanced deep learning algorithms have been proposed to achieve higher detection accuracy. However, those systems generally involve high latency, require considerable memory space, and often result in high energy consumption. To accelerate intrusion detection and also reduce memory and energy costs, we propose a new IDS system using Binarized Neural Network (BNN). Compared to full-precision counterparts, BNNs can offer faster detection, smaller memory cost, and lower energy consumption. Moreover, BNNs can be further accelerated by leveraging Field-Programmable Grid Arrays (FPGAs) since BNNs cut down the hardware consumption. The proposed IDS is based on a BNN model that suits CAN traffic messages and takes advantage of sequential features of messages rather than each individual message. We also explore various design choices for BNN, including increasing network width and depth, to improve accuracy as BNNs typically sacrifice accuracy. The performance of our IDS is evaluated with four different real vehicle datasets. Experimental results show that the proposed IDS reduces the detection latency (3 times faster) on the same CPU platform while maintaining acceptable detection rates compared with full-precision models. We also examine the proposed IDS on multiple platforms, and our results show that using FPGA hardware reduces the detection latency dramatically (128 times faster) with lower power consumption compared to an embedded CPU device. Furthermore, we evaluate BNNs with different designs. Results demonstrate that wider or deeper models definitely improve accuracy at the cost of increased latency and model sizes to varying degrees. Applications are recommended to choose the appropriate model design they need depending on available resources they have.

INDEX TERMS Automotive security, intrusion detection, in-vehicle network, controller area network (CAN), binary neural networks, machine learning.

I. INTRODUCTION

With the rapid development of new mobility services as well as autonomous and connected vehicle technologies, the transportation industry is experiencing a revolutionary transformation. New types of mobility promise to virtually eliminate some chronic problems in the current landscape of the automotive world, such as crashes and fatalities. To support emerging mobility services, more software modules and external interfaces are added to modern vehicles, and in-vehicle networks can communicate with the outside world. As in-vehicle networks are no longer in isolated environments, new attack surfaces emerge [1], [2], [3], [4], [5]. Consequently, in-vehicle networks can be vulnerable to various cyber-attacks which may result in serious consequences. Automotive manufacturers have to first eliminate malicious

The associate editor coordinating the review of this manuscript and approving it for publication was Jad Nasreddine.

actors and minimize their impact in order to deliver these promising and advanced services. Automotive cybersecurity has become a significant problem in today's automotive industry.

A modern automobile has tens to over a hundred of Electronic Control Units (ECUs) which are connected by the in-vehicle network to control electrical subsystems [2], [6]. Researchers have demonstrated that attackers can intentionally control a target victim vehicle via a malicious node connected to the in-vehicle network [2], [6], [7], [8]. Those attacks show that the security of in-vehicle networks becomes a critical concern as it closely relates with the safety of drivers and passengers. Attackers are able to access Controller Area Network (CAN) (which is the most predominant in-vehicle bus communication protocol) through Bluetooth, Wi-Fi or other surfaces. Since CAN has no security features by design, such as encryption and authentication, attackers can exploit CAN vulnerabilities to launch vehicle cyber attacks. For example, they can control subsystems of a victim vehicle by injecting malicious messages via the on-board diagnostic (OBD-II) port physically or wireless communication systems remotely. Attackers are able to further compromise normal ECUs by reprogramming their firmware, and then they can manipulate the compromised ECUs to interrupt the normal operations by sending arbitrary CAN messages or even control the vehicle [4], [6], [9], [10], [11], [12].

Several studies attempt to protect in-vehicle networks by encryption and authentication approaches to ensure the integrity and confidentiality of transmitted data frames. However, due to the very limited space in the data frame (a CAN message has 8-byte data at most) and the demanding real-time requirement of CAN bus, it is still a challenging job to have a practical deployable solution for encryption and authentication. Parallel to encryption and authentication approaches, intrusion detection system (IDS) is considered an effective approach that can protect in-vehicle networks by detecting malicious messages [3], [13]. Advanced machine learning techniques, such as Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) [13], [14], [15], have been used to create models for IDSs to identify attacks. Neural networks have shown remarkable capabilities in complicated applications including image classification, semantic segmentation and object detection [16], [17]. However they generally suffer from high computational complexity, usually require considerable computing resources, and often result in large power consumption. A neural network model can require hundreds of Megabytes (MBs) of memory [18], [19], [20], [21]. This requirement for memory accesses is often the bottleneck of system performance as well as energy efficiency [22]. Therefore, it can be difficult to deploy neural networks on resource-constrained devices in embedded environments. Considering the constraints of ECUs' memory and the demand of real-time communications of in-vehicle networks, an IDS for in-vehicle networks should be small in size and have a fast responding time during the detection process.

To overcome those limitations of deep-learning based IDSs and protect CAN against vehicle cyber attacks, we propose a new IDS based on Binarized Neural Network (BNN) that uses binary values for activations and weights instead of full-precision values. The BNN model can accelerate intrusion detection and reduce memory requests as well as energy consumption compared with the corresponding full-precision neural networks.

To allow our BNN model to learn and process CAN traffic for intrusion detection, we create an input generator to assemble CAN data to input frames that can be processed by our BNN model. Since this input generator converts bit-stream data of CAN traffic to a grid format rather than a feature vector format, the BNN model can learn the temporal sequential pattern without the need for extra data preprocessing. Moreover, besides the acceleration offered by BNN, our scheme can be further accelerated with Field Programmable Gate Arrays (FPGAs), a promising hardware acceleration technique. FPGAs implementation benefits from converting floating-point multiplication operations to inexpensive bit-shift operations. Due to the low power consumption and high flexibility compared with CPUs and GPUs, FPGAs implementation meets the demands of embedded systems especially. However, current FPGAs cannot be used to accelerate complex neural network models as FPGAs cannot offer large enough computing and memory resources. Since BNNs have bitwise activations and weights, our BNN-based IDS is able to be further accelerated by FPGAs [23], [24], [25], [26], [27], [28], [29], [30], [31].

BNNs are compact and efficient due to binarized weights and activations. However, accuracy is usually sacrificed to some degree. Improvement strategies can be applied to mitigate the loss of accuracy [32], [33]. As our goal is to utilize BNNs with higher accuracy while maintaining relatively low latency and memory cost, we explore various design choices of BNNs, including increasing network width and depth methods, to investigate if and how those methods can improve accuracy.

To evaluate the proposed model, we collect datasets from four real vehicles from different car manufactures. Our experimental results show that the proposed IDS can achieve satisfying detection rates as well as low latency. We also test our IDS on FPGAs and other platforms to show the performance of accelerated inferences, and we evaluate the memory utilization, power consumption on different platforms, including CPUs, GPUs and FPGAs. Moreover, we investigate BNNs with different design choices to see how they can improve accuracy performance. Our experimental results demonstrate that wider and deeper models provide better accuracy performance at the cost of increased latency and model sizes to varying degrees.

This paper makes the following contributions:

- To the best of our knowledge, this is the first work applying BNN to IDS for in-vehicle networks. We adopt the BNN model [34], [35] and convert CAN bit-stream

traffic to the sequential pattern of CAN messages to feed into our BNN model.
- Our BNN-based IDS has lower latency (3 times faster) compared with another full-precision NN-based IDS. It also requests smaller memory size as well as lower power consumption. Small model size and low power consumption are important for in-vehicle network environments as they consist of resource-constrained embedded devices. Moreover, the proposed BNN-based IDS can be further accelerated by executing on FPGA hardware (128 faster than on an embedded CPU).
- We perform comprehensive evaluations with real CAN traces collected from four real vehicles, while most previous studies either use simulated data or data collected from no more than two vehicles.
- We evaluate and analyze various BNN models to determine how the width and the depth of a BNN can affect performance. Our experiments show that both deeper or wider models can offer higher accuracy. Different applications can choose different models according to their specific needs and available resources.

A preliminary version of this article appeared in [36].

Organization. This paper is organized as follows. Section II provides an overview of CAN and its security risks. Section III reviews related work. Section IV details the adversary model. Section V provides the design of the proposed IDS. Section VI details the insights into the implementation and evaluation metrics of the proposed scheme. Section VII discusses the proposed IDS. Finally, we conclude the paper in Section VIII.

II. BACKGROUND

A. CONTROLLER AREA NETWORK

As a message broadcast bus, CAN provides reliable interconnections among ECUs by a multi-master broadcast serial bus system [37], [38], [39], and it is the most widely used in-vehicle communication protocol.

1) CAN MESSAGE

Each CAN message can carry up to 64-bit data. CAN messages transmitted in the bus maintain data consistency and deliver information among ECUs regarding making control decisions. As shown in Figure 1, the format of a CAN message includes: Start-of-Frame (1 bit), Identifier (11 bits or 29 bits), Control Field (6 bits), Data Field (0-8 bytes), CRC Field (16 bits), ACK Field (2 bits), End-of-Frame (7 bits) and Inter-Frame Space (3 bits). Note that there are two message formats depending on the number of identifier bits: base frame format (with 11-bit ID) and extended frame format (with 29-bit ID). Based on the CAN standard, the implementation of CAN must accept the base frame format and must tolerate the extended frame format. Besides this original CAN, CAN-FD, an extension to the original CAN, extends the data field to up to 64 bytes. This work focuses on the classical CAN.

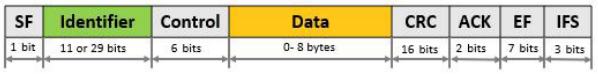

FIGURE 1. CAN message format.

2) MESSAGE BROADCAST

CAN bus is a message-oriented network, and CAN messages have no addressing scheme. Each frame is assigned a CAN ID instead of relying on sender and receiver address information to communicate. Each ECU is configured to accept CAN messages with specific CAN IDs and disregard others at compile time. CAN bus gets flexibility as operations (such as adding, replacing or removing an ECU node) do not affect other nodes. When a CAN message is broadcast, all ECUs in the bus can receive it. If the message is not on a predefined receiving list of a certain ECU, that ECU will discard it. Otherwise, the message will be accepted and further processed. Figure 2 shows a typical CAN bus data transfer process. ECU 1 broadcasts a message, then both ECU 2 and ECU 3 receive the message. ECU 2 checks the message ID and discards it because the ID is not recorded in its specifications. ECU 3 accepts the message after verifying the ID as it was configured to accept messages with that ID.

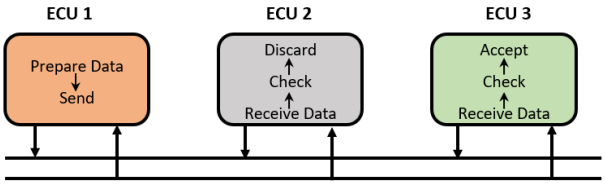

FIGURE 2. CAN Bus topology.

3) CAN BUS ARBITRATION

As a broadcast communication network, CAN uses message priority for collision detection and arbitration. When an ECU prepares to send a message, the ECU needs to check the bus status. The message will only be sent if the bus is idle. Otherwise, the ECU has to wait for the next opportunity. CAN arbitration solves collisions where multiple messages are sent at the same time. The arbitration is based on the priority of CAN messages defined by their IDs, and the message with the smaller ID value has the higher priority [37], [38], [39], [40]. For example, when three CAN messages with ID 0x0001, 0x022 and 0x00FF are sent to the idle bus simultaneously, the message with ID 0x0001 can dominate the bus and broadcast itself. The other two messages with the lower priority have to wait for the next time. Car manufacturers decide the allocation of the message priority.

4) TYPICAL CAN SETUP

A typical CAN setup is that each ECU sends messages with a unique set of IDs. Each message from an ECU is assigned with an ID from the unique set of that ECU. Messages with the same ID must be sent from the same ECU, because the unique set of IDs for each ECU is non-overlapped with each

other. Similarly, each ECU is also configured to accept and process messages with a set of message IDs. However, those sets can be overlapped among different ECUs. As a result, a CAN message can only be sent from one ECU, but it can be accepted and processed by multiple ECUs.

B. CAN VULNERABILITIES AND ATTACKS

Even though CAN physical layer has strong error detection through CRC, bit stuffing, etc., it has no security protection. Most of CAN vulnerabilities come from those facts: it is a broadcast bus; all messages are broadcast to all nodes; each message has an ID without sender information, and each node decides if it should process the message. Such a design introduce vulnerabilities for the CAN communication: 1) all nodes see the whole traffic, which allows eavesdropping and learning patterns of certain ECUs; 2) any node deployed on the CAN bus can send any arbitrary message and no one knows who send that message; 3) the broadcast nature and relying on arbitration to win bus access make DoS attack possible; 4) answers to standard challenges that are needed for authentication when doing sensitive things, such as reflashing components or firmware update, are stored in memory, etc. Those vulnerabilities provide opportunities for attackers to mount attacks for certain purposes. For example, due to the lack of sender ECU identification, the attacker who compromises an ECU can send malicious messages that can disrupt the bus and even cause control system failure without leaving a trace.

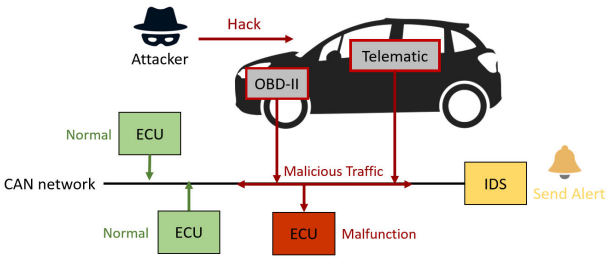

FIGURE 3. Typical CAN bus attack scenario.

Figure 3 shows a general attack procedure for the CAN bus. The procedure can be divided into three phases [5], [41], [42]: investigation phase, preparatory phase and attack phase. In the investigation phase, to access the target in-vehicle network, attackers need to determine an interface that can be leveraged, such as the OBD-II port, the telematics system (e.g., Ford's Sync and GM's OnStar), etc. Then, attackers can build a malicious node through the chosen interface. The malicious node can be an *external* device (such as a laptop, an external ECU) or an *internal* ECU (such as a compromised ECU, a telematics system infected by malware). In the preparatory phase, attackers leverage the malicious node to eavesdrop and analyze CAN messages transmitted on the bus. All transmitted CAN messages can be eavesdropped and recorded because of the broadcast nature of CAN. Based on that, attackers are able to further analyze historical CAN messages and then design malicious messages with certain purposes. In the attack phase, adversaries can implement various types of attacks based on their design, such as injecting malicious messages. It is noted that malicious messages for a specific victim vehicle cannot be reused to attack other vehicle models, because the implementation of CAN varies by model.

C. INTRUSION DETECTION SYSTEM

An IDS protects a network from attacks or malicious activities by monitoring the traffic of the network. When an intrusion is detected, an alarm will be raised. Machine learning algorithms have been used extensively as a powerful mathematical tool for various tasks, such as classification, regression, and clustering. As intrusion detection is a classification task, those algorithms can be used to build classifiers and can be developed as IDSs. In the automotive environment, IDSs classify the currently transmitted messages in two categories: normal or malicious, which is a well-known binary classification task. Those machine learning-based IDSs generally need more computing resources, and their detection processes may be time-consuming. However, advanced machine learning algorithms, such as neural networks, have unique advantages in providing accurate results. Researchers have proposed various IDSs based on machine learning algorithms for in-vehicle networks, including ANN, DNN and CNN, etc.

III. RELATED WORK

CAN security becomes more critical with the recent advancements in the automotive industry and the introduction of autonomous and connected vehicle technologies. Today's vehicles are no longer isolated mechanical systems as electronic controls and x-by-wire systems can control almost every aspect of vehicles. A driver will not be required to control the car, and every car is going to be connected and communicate to other cars and other things (V2X). These developments make automotive cybersecurity become a critical topic. In order to evaluate the capability of attackers, several studies have demonstrated that attackers are able to access the CAN network via a variety of attacking interfaces, such as the Tire Pressure Monitoring System (TPMS), Bluetooth, telematics, and OBD-II, and intentionally launch attacks or even take the whole control of the victim vehicle [4], [6], [9], [10], [11]. Miller *et al.* present the attack process that attackers can control a driving Jeep Cherokee on highway by remotely manipulating compromised ECU(s) [6]. Nie *et al.* mount several remote attacks on the Tesla Model S/X in both parking and driving mode [4], [9]. Their studies show that attackers are able to access the Autopilot ECU(s) in the vehicle and execute remote attacks through exploiting vulnerabilities.

A. INTRUSION DETECTION SYSTEM FOR CAN

IDS is one of the major defense mechanisms commonly used to secure CAN from attacks. Most IDSs are based on anomaly detection where the IDS identifies attacks by comparing normal behaviors with abnormalities. Depending on

the different detection methods used, IDSs for CAN can be categorized into two main categories for anomaly detection: 1) rule-based; 2) machine learning-based. Rule-based IDSs exploit different characteristics of CAN messages. Some proposals leverage the fact that most CAN messages are sent periodically with fixed intervals [3], [43], [44], [45]. Schemes proposed in [43], [45], and [46] apply entropy-based methods to detect intrusions by monitoring the changes in intervals and entropy. Abnormal changes in system entropy and relative entropy are counted as intrusions. Song *et al.* detect message injection attacks by analyzing traffic anomalies based on an assumption that *all* CAN messages are generated at regular frequency or interval [44]. The clock-based IDS proposed in [3] uses the periodical nature of many CAN messages to detect anomalies as well as to fingerprint ECUs. The scheme proposed in [47] measures device-specific delay time as a mechanism for anomaly detection. In particular, the delay is measured with a resolution that is implemented on FPGAs as measurement using software may miss messages when processing. Although lightweight, all these approaches may not be effective for attacks with aperiodic messages. In summary, the aforementioned schemes can offer fast response and usually work well with specific threat models. However, they generally cannot cover all possible types of attacks. They cannot detect attacks that are not included in the specific threat models.

As cars become more connected and complicated, new and sophisticated attacks are emerging. Machine learning methods, such as Support Vector Machines (SVM), random forests and decision tree, have been introduced to design CAN IDSs [48], [49]. However, IDSs based on these classical machine learning algorithms, including both general IDSs [15], [50], [51], [52] and CAN IDSs [13], [14], generally cannot offer high enough accuracy. For better performance, advanced data analytic techniques such as deep neural networks are considered to improve the detection rate. They are able to detect more sophisticated attacks or unknown attacks that can evade detection methods relying on regularity, periodicity, or other data characterises, such as simple sequence patterns. Machine learning-based IDSs for in-vehicle networks have been proposed [1], [13], [14], [53], [54], [55], [56], [57], [58]. These schemes use different machine learning algorithms as well as various data features to train their model for intrusion detection. Wasicel *et al.* [55] propose an IDS based on an artificial neural network that uses data extracted from the OBD-II port to train the machine learning models to detect anomalous activities in vehicles. The data used have clear semantic meaning, which adds delay in data processing. The authors of [56] propose a vehicular intrusion detection system based on Long Short-Term Memory (LSTM), a type of RNN, which takes time frequency difference between CAN messages as input. Kang *et al.* [14] present an IDS by applying the bit pattern of the data field (64-bit) as feature vectors for DNN. Their results show that the DNN-based approach outperforms traditional ANN-based approaches on the detection accuracy. Song *et al.* propose a CNN-based IDS using the sequential pattern of the CAN traffic [13]. They use every 29 consecutive CAN IDs to build frames for both training and testing. However, since their IDS only covers the ID field of CAN messages, it has no ability to detect malicious messages with the valid ID and malicious data. Zhang *et al.* propose a hybrid IDS consisting of two stages. The first stage is used to quickly detect attack messages that violate the established rules while the DNN-based second stage is used to detect attack messages that fall out of the scope covered by the first stage. Experimental results show that the proposed hybrid framework can achieve advantages of both rule-based and machine learning-based approaches [59], [60].

B. INTRUSION DETECTION MODEL BASED ON SEQUENTIAL PATTERN

Machine learning techniques have proven to be successful at conducting intrusion detection throughout the years, and intrusion detection using sequential pattern is one of the research directions. Oliveira *et al.* propose an approach by using sequential pattern of data, and their results show that anomaly detection can be better addressed from a sequential perspective [61]. Xing *et al.* categorize three types of classification methods based on the sequence pattern of data [62]. The first one is the feature-based method that transforms sequences to feature vectors through feature selections, and then applies classification methods. This method introduces an additional preprocessing step before the classification. The second method is the distance-based method that examines the similarity among sequences. This method may not work effectively in environments that involve a large-scale database as it is difficult to calculate distances of all reference sequences. The last method utilizes machine leaning algorithms, such as naive bayes, markov model, hidden markov model and neural networks, to build sequence models for sequences classification. However, for high accuracy, they usually involve high computation cost and require large amount of data in the training process. Wang *et al.* propose a malicious traffic classification method using CNN [63]. By transforming the raw network traffic data as images for their training and testing of the CNN classifier, there is no necessity to introduce any hand-designed features or feature selections. To protect CAN in particular, with the similar deign, an IDS based on CNN is proposed [13]. CAN traffic data are converted to image-like frames with grid structure rather than feature vectors, so therefore their CNN classifier is able to learn the sequential patterns of CAN traffic to detect intrusions. Their experiments demonstrate that the proposed IDS has low false negative rates and error rates.

C. ACCELERATING NEURAL NETWORK

1) BINARIZED NEURAL NETWORK

Due to the high computation cost and the large model size, advanced neural network techniques can be difficult to deploy in embedded environments. Some studies aim on making deep learning faster and smaller without sacrificing overmuch accuracy. Han *et al.* [64] discuss that neurons not

contributing much to the network can be removed from the network. Pruning network is sparser and potentially smaller with fewer calculation. Iandola *et al.* [65] present a method to reduce the number of parameters for network size reduction. Both methods aforementioned can make the model size smaller to some extent, but they cannot guarantee a big size reduction when most neurons in the network significantly contribute to the network or the model is not over-parametered. Researchers [35], [66] propose a quantized network in the extreme case: Binarized Neural Network. They introduce a method to train neural networks with binary weights and activation. The result of BNNs is obvious [66], in comparison with 32-bit DNNs. BNNs can require up to 32 times smaller memory size and 32 times fewer memory accesses. However, BNNs may not be able to provide higher accuracy compared to their corresponding full-precision models. Various of methods have been reported to improve BNN accuracy [33]. Some studies focus on improve model structure. Bulat *et al.* [33] design a hierarchical network and suggest make each layer wider by adding more neurons in hidden layers. Tang *et al.* [67] propose to use partial binarization during training, and only binarizing groups of kernels that have a greater impact on overall performance. Another direction of improving accuracy of BNNs is to extend core principals of binarization itself, such as scaling with a gain term. Gain terms [31] can be used to give more capacity to a network when multiple gain terms are used within a dot product or to form a linear combination of parallel dot products.

2) DEEP LEARNING ON FPGAs

Deep-learning applications with demanding real-time responses rely on parallelism computing in inference phases [29]. In the early years, since GPUs were specifically designed for videos and images rendering, using GPUs for deep learning became well-accepted. GPUs are able to process numerous arithmetic operations in parallel so that they can offer considerable acceleration. However, comparing with Application-Specific Integrated Circuits (ASICs), which are specially optimized for deep learning applications, GPUs do not deliver as much energy efficiency as ASICs at the same performance. While there is no single hardware architecture working perfectly for all deep learning applications, FPGAs provide distinct advantages over GPUs and ASICs in certain use cases. FPGAs offer flexibility and cost efficiency with circuitry that can be reprogrammed for desired functionalities. Comparing with GPUs, FPGAs provide decent performance in deep learning applications in which low latency and high power efficiency are critical. Comparing with ASICs, FPGAs can be fine-tuned even after being manufactured to set a balance of power efficiency and performance with specific requirements. As accelerator devices, FPGAs can be used in the deep training phase, which has already been widely deployed in the cloud server by leading technology companies, such as Microsoft Azure [68]. It is also feasible to use FPGAs as embedded devices in the inference phase. In our case, we propose to use an FPGA device as an in-vehicle IDS deep learning accelerator. Unlike powerful and expensive FPGAs in cloud servers, embedded FPGAs have limited logic resources and memory bandwidth, which makes them difficult to perform a full-size, full-precision deep learning inference [27]. To bridge this gap, one of the approaches [69], [70] is to optimize deep learning computation and memory access in FPGAs by efficiently increasing the utilization of the design space. Another approach [71] is to simplify the deep learning model by reducing the precision of floating-point operations and fit this size-reduced model onto an embedded FPGA device.

IV. ADVERSARY MODEL

We consider general adversary models which can cover most known attack scenarios targeting the CAN communication. In this work, we assume that the attacker already has access to the CAN bus of the target victim vehicle and can manipulate the malicious node to launch the attack by injecting malicious messages. The malicious node can be either an external device, such as a laptop, an external ECU connecting to the bus, or an internal ECU, such as a compromised ECU [2], [3], [4], [6], [7], [8], [9], [11], [41]. Due to the broadcast nature of the CAN communication, malicious CAN messages injected by attackers are broadcast on the bus, and other ECUs cannot distinguish who the sender is.

Based on the adversary's different levels of capabilities and different content of CAN messages, attacks can be classified as follows:

A. RANDOM ID ATTACK

When an attacker does not understand the semantic meaning of CAN messages of the target victim vehicle and has no possession of previous CAN traffic traces, he can only launch attacks by using simple messages. In this attack, the attacker generates random CAN ID messages and injects them into the CAN bus. The attacker does not need to have prior knowledge of the target vehicle. The objective of this attack is to disturb normal functions of certain subsystems or to compromise the usability of normal CAN messages.

B. ALL ZERO ID ATTACK

When an attacker considers the priority of CAN messages and CAN arbitration, he can launch attacks using all zero ID messages. Those messages have the highest priority as the lower ID value represents the higher priority. The attacker aims to occupy the bus and achieve a DoS (denial-of-service) effect resulting in severe safety consequences.

C. REPLAY ATTACK

When an attacker has previous CAN traffic data, he can mount attacks by replaying those messages. In this attack, the attacker injects valid messages transmitted in the bus before to the bus. This attack can be realized easily, because CAN bus lacks freshness protection mechanisms.

D. SPOOFING ATTACK

When an attacker can access current or previous traffic traces or has knowledge about the specification of certain CAN

IDs either through reverse engineering or learning from other sources such as the proprietary Database Container (DBC) file, also known as CAN database files, he can spoof certain functions by using spoofed messages to harass the CAN bus or control the vehicle behaviors. In this attack, the attacker injects spoofed messages pretending to be from a valid ECU. To create a valid spoofed message, the attacker needs to understand the messages sent from the victim ECU. This attack can allow the attacker to intentionally control the target vehicle which results in critical safety issues.

V. OUR DESIGN

Our objective is to build an in-vehicle network IDS to protect the CAN bus by detecting suspicious or malicious activities in the bus. Meanwhile, this IDS aims to have low latency on the detection process. It is supposed to be placed on a central gateway or installed on an ECU as a node of the CAN bus that can monitor the CAN traffic.

We propose a new IDS using BNN to identify intrusions in order to overcome limitations on existing machine learning-based IDSs and protect in-vehicle networks against vehicle attacks. BNNs are types of neural networks using one-bit quantization for weights and activations, and they can provide shorter inference time and smaller memory requests than corresponding full-precision neural networks. The actual IDs and data of a vehicle's CAN messages depend on the manufacturer's design, and this information (usually defined in manufacturer's proprietary DBC file) is kept strictly and not published for security reasons. We use raw CAN messages that are transmitted on the CAN bus and leverage the sequential patterns of messages to detect malicious intrusions without the need for DBC files.

As illustrated in Figure 4, the proposed BNN-based IDS consists of two steps: 1) the training step and 2) the detection step. In the training step, we use labeled inputs for training. It can be challenging for the BNN model to directly learn CAN messages. Therefore, we design an input generator that assembles ten consecutive CAN messages and attaches corresponding labels to build input frames. If one or more attack messages are in one input frame, the frame label is set as malicious. Otherwise, the label is set as normal. In this way, BNN is able to learn sequential patterns during the training process. Note that training can be performed offline as training a classifier is considered time-consuming. Once the training process is completed, the trained BNN model is used in the detection process. In the detection step, the input generator only assembles current traffic messages as inputs without attaching labels. Those inputs are processed by the trained BNN model where the topology and parameters are learned from the training step. The model can be deployed in general-purpose hardware such as CPUs, GPUs or FPGAs. The output of the model is a prediction of whether the input is malicious or not. If an input is identified as malicious, an alarm will be raised.

A. INPUT GENERATOR

Neural networks are able to take raw data as inputs without any extra hand-designed features. Our goal is to process raw

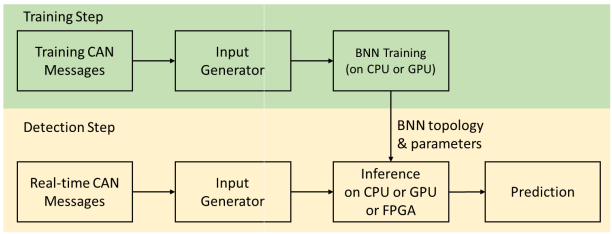

FIGURE 4. Workflow of the proposed IDS.

CAN traffic data without feature engineering efforts that also save the corresponding processing time.

As described in Section IV, an attacker can launch various attack types by injecting different types of malicious messages. Figure 5 provides two CAN traffic log segments without and with attacks. Each line shows one message. Both "normal" and "attacks" segments are collected for 7 ms. In normal conditions, the number of message with ID 064 occurrences is 2. However, when there is an attack, the number of message occurrences increases to 9 messages. As a result, the sequential pattern of the CAN traffic is changed, which can be exploited to detect attacks.

```
ID: 12C   Data: 7B 72 00 00 02 EB 11 CC
ID: 064   Data: FF C0 00 00 EE 00 8A CC
ID: 188   Data: C6 06 00 02 00 00 0E CC
ID: 090   Data: 02 04 FF C1 00 0F 08 DB
ID: 1C0   Data: 00 00 00 00 00 00 00 00
ID: 0D4   Data: 00 E0 00 48 56 00 00 CF
ID: 0A6   Data: 00 00 09 0A 00 00 00 0D
ID: 0C8   Data: 00 00 00 3B 00 57 00 06
ID: 12C   Data: 7B 72 00 00 02 E6 11 0D
ID: 188   Data: C6 06 00 02 00 00 0E 08
ID: 20C   Data: 02 54 00 00 02 E6 00 87
ID: 0D4   Data: 00 E0 00 48 56 00 00 0B
ID: 1F4   Data: 04 64 00 01 3D F7 00 80
ID: 0A6   Data: 00 00 09 0A 00 00 00 49
ID: 0C8   Data: 00 00 00 3C 00 57 00 41
ID: 064   Data: FF C0 00 00 EE 00 8A 08
ID: 12C   Data: 7B 72 00 00 02 E6 11 49
ID: 090   Data: 02 05 FF C1 00 0F 03 1B
```

```
ID: 0C8   Data: 00 00 01 7E 01 90 00 C4
ID: 064   Data: FF C0 00 00 EE 00 65 4B
ID: 12C   Data: 78 71 00 00 02 EA 10 C2
ID: 090   Data: 02 05 00 27 00 0F 0F 5D
ID: 188   Data: 21 01 80 08 00 00 0E C2
ID: 1C0   Data: 03 1C 06 50 0C 20 19 67
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 0D4   Data: 00 E0 00 45 65 00 65 C7
ID: 0A6   Data: 00 00 09 09 00 00 00 0E
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 0C8   Data: 00 00 01 81 01 94 00 08
ID: 12C   Data: 78 71 00 00 02 EA 10 0E
ID: 188   Data: 21 01 80 08 00 00 0E 0E
ID: 20C   Data: 02 54 00 00 02 EA 00 0B
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 0D4   Data: 00 E0 00 45 65 00 65 03
ID: 1F4   Data: 04 64 00 01 3D F7 00 C0
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 064   Data: FF C0 00 00 EE 00 8A 00
ID: 2E4   Data: 57 00 00 00 00 00 00 8C
ID: 0A6   Data: 00 00 09 09 00 00 00 4A
ID: 0C8   Data: 00 00 01 83 01 97 00 4F
ID: 064   Data: FF C0 00 00 EE 00 65 87
```

FIGURE 5. CAN traffic log without attack (left) and with attack (right).

To enable the BNN model to exploit the sequential pattern of CAN traffic, we design an input generator that transforms consecutive CAN messages to grid structure frames, and then the BNN model learns the sequential pattern of these input frames. Each input frame contains data bits extracted from N consecutive messages and padding bits. In our design, the generator extracts 11 bits from the ID section and 64 bits from the data section, and the bit representation of each message is as follows:

$$M = b_0^{id} \cdots b_{10}^{id} b_0^{data} \cdots b_{63}^{data} \tag{1}$$

We conduct experiments to find the proper size N of the input frames and report details in Section VI-D. Based on experiments, we set $N = 10$, i.e. the input generator assembles every ten sequential CAN messages into an input frame. Figure 6 shows an example input frame. In the last ten consecutive CAN messages, the generator extracts 75 bits from each message (11-bit ID section and 64-bit data section) for a total

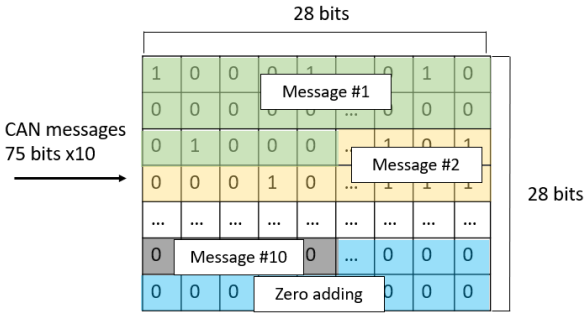

FIGURE 6. An example input frame generated by the input generator.

of 750 bits from ten messages, plus 34 bits for padding. Then, the generator builds a 28×28 bitwise frame by stacking these 784 bits. By utilizing this input generator, no hand-designed features are needed, and our IDS can also be more efficient through batch processing compared to other IDSs [59], [60], because multiple CAN messages in a frame are processed at once during detection.

Many CAN IDSs [13], [44], [72] only rely on the ID field of CAN messages for intrusion detection. Therefore, they are ineffective to detect malicious messages that contain valid ID and malicious data content, such as spoofed messages. Unlike those schemes, our IDS learns from input frames that include full information of each message and is able to detect such attacks.

Using the signal data (0 or 1), rather than decoded values, does not introduce extra prepossessing. Considering that thousands of messages are transmitted per second on a CAN bus, such computational efficiency should be taken into account for an IDS design. Furthermore, neural networks generally need to normalize data to improve accuracy, because a dominating feature with a bigger scale compared with other features causes low accuracy. Our model does not need normalization for input data, which also further improves efficiency by avoiding the computation cost on normalization as all inputs consist of only 0 or 1. In addition, the 0 or 1 data helps our IDS map to FPGA hardware more efficiently because these data make bit manipulation easier for the hardware.

B. BINARIZED NEURAL NETWORK

Unlike common neural networks which calculate the parameters in floating-point format, BNNs use binarized weights and activations instead of full-precision ones. By using bitwise operations, BNNs can substantially reduce computation cost and improve efficiency. In full-precision neural networks, the basic operations usually can be expressed as Equation 2, where z is the output tensor and σ represents a non-linear function. w and a represent the weight tensor and activation tensor, and their outer product is decoded as $\otimes$.

$$z = \sigma(w \otimes a) \tag{2}$$

In BNN, during the forward propagation stage, floating-point weights w and activations a are replaced by w_b and a_b that are all 1-bit long, which can be defined as follows:

$$Q_w(w) = \alpha w_b, Q_a(a) = \beta a_b \tag{3}$$

A BNN can be reformulated in the regular DNN format as follows:

$$z = \sigma(Q_w(w) \otimes Q_a(a)) = \sigma(\alpha\beta(w_b \odot a_b)), \tag{4}$$

where $\odot$ is an inner product for vectors that operate using XNOR-Bitcount [73].

1) BINARIZATION IN BNN

The essential idea of BNN is to set weights and activations to $+1$ or -1 [34], [35]. Two binarization methods have been introduced: stochastic or deterministic. We use the deterministic method as it does not require hardware to generate random bits during quantization. Since the output from our input generator is either 0 or 1 without any normalization process, the first input of BNN and the rest of layers can be both formulated as Equation 5:

$$x^b = Sign(x) = \begin{cases} 1 & \text{for } x \geq 1 \\ -1 & \text{for } otherwise \end{cases} \tag{5}$$

The problem is that the derivative of the sign function gets almost zero everywhere while in the back propagation process. [34] uses Straight-Through Estimator (STE) in a deterministic way, which is called hard tanh (*HTanh*):

$$HTanh(x) = \begin{cases} 1 & \text{for } x \geq 1 \\ x & \text{for } -1 < x < 1 \\ -1 & \text{for } x \leq -1 \end{cases} \tag{6}$$

Using STE, the 32-bit floating-point weights, denoted as W_{32bit}, are updated with an optimization strategy. During a large portion of training, a positive value of W_{32bit} is evaluated to have a positive gradient, and that value is increased in every update. If values in W_{32bit} are not bounded, they will be accumulated to large numbers. For this reason, BNNs clip the values of W_{32bit} between -1 and $+1$. This keeps the values of W_{32bit} closer to binarized weight W_{1bit}.

2) MODEL DESIGN

We design a simple but powerful fully-connected network. As shown in Figure 7, this network takes input frames (each frame consists of ten consecutive CAN messages) from the input generator and outputs a binary decision indicating whether or not there are any malicious messages among the ten messages. We set three hidden layers for the network and 1024 nodes for each hidden layer as a baseline BNN model. In each hidden layer, there are three sub-layers: 1) a binarized dense layer. It is deeply connected with its preceding layer. All weights and activations are binary values, except for the first binarized dense layer where only weights are binarized;

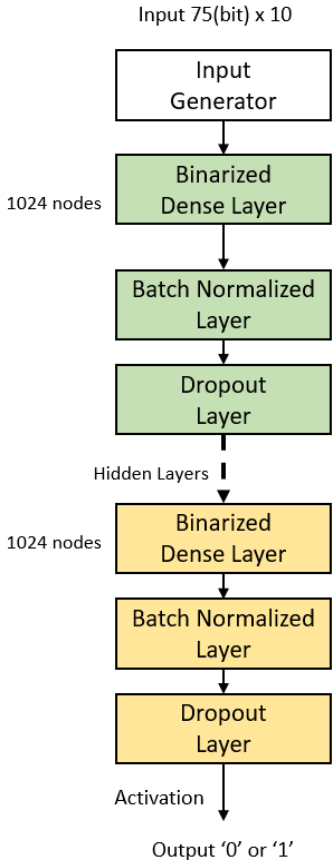

FIGURE 7. Baseline BNN structure.

2) a batch normalization layer. It standardizes the inputs to a layer for each mini-batch, which helps prevent overfitting; 3) a dropout layer with a 15% dropout rate. It is to reduce overfitting and improve the generalization of deep neural networks.

We adopt training strategies described in [66] including shift-based batch normalizing and AdaMax. To train our model, and make it more efficient to be mapped to FPGAs without affecting the network accuracy, additional optimizations are applied:

- Converting the input frames to 0 or 1 through the input generator makes it easier for hardware to process the bit-wise operations. Then 1-bit values are used for all input activations, weights, and output activations to make the model full binarization, where an unset bit represents -1 and a set bit represents +1.
- Based on [30], the summation of a binary dot product can be implemented on hardware by a popcount operation where the set bits are counted by avoiding signed arithmetic accumulation.

In the detection step, almost all parameters in the BNN are binarized during the inference stage. There are 3.95 million multiply-accumulate operations, but 3.12 million of them are binary multiply-accumulate, which makes the network work fast if it is deployed on custom BNN hardware.

3) BNN INFERENCE ON FPGAs

While a BNN model is trained on powerful computational devices, mostly on CPUs or GPUs, the trained model is able to be deployed on general hardware. Previous studies [29], [30], [31] show that the BNN can work on resource-constrained devices, such as embedded CPUs, ASICs or FPGAs. FPGAs are considered as one of the most widely used platforms for BNNs. FPGAs offer hardware customization and can be programmed to deliver performance similar to GPUs or ASICs. The reconfigurable nature of FPGAs lends itself well to the rapidly evolving AI landscape and marketing requirement. In this work, we utilize an FPGA inference framework from Xilinx [35], which is originally designed for computer vision tasks, such as image classification. Because the input generator transforms the CAN traffic to input frames, we make our BNN-based IDS also work on this framework while enjoying the further acceleration from the FPGA during the inference. Figure 8 illustrates the general workflow of converting a trained IDS into an FPGA accelerator. IDS supplies frames and a trained BNN to Finn synthesizer. The synthesizer first determines the resource allocation by Matrix–Vector–Threshold Unit (MVTU), a computational core for the accelerator hardware design, to meet the frames per seconds (FPS) target and applies the optimizations. And then it produces a synthesizable network description of a heterogeneous streaming architecture [30], an FPGA parallel optimizing design. In the last step, with the hardware library and Vivado HLx design suite, the data stream can be processed on the target FPGA platform.

Although the size of a full-precision neural network may be practically implemented on some advanced FPGA hardware, a smaller model size is always beneficial for CAN IDSs, given the constraints of ECU memory, the need for real-time communication of in-vehicle networks and budget friendly considerations [74] (advanced FPGAs can be high-cost). Our BNN-based IDS takes advantage of neural networks and can accelerate CAN intrusion detection with a lower power consumption, a smaller model size, as well as lower budgets.

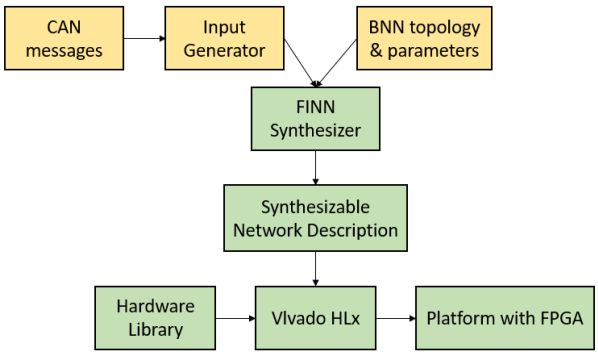

FIGURE 8. FPGA inference workflow.

C. ENERGY CONSUMPTION AND MEMORY COST

Improving computing performance is still challenging for neural networks on various tasks. Power consumption has

TABLE 1. Details of datasets.

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Vehicle Model	Honda Accord 2006	Honda Civic 2018	Ford Fusion 2013	Chevrolet Volt 2013
Number of Messages	243,762	1,806,780	2,158,201	4,536,342
Number of Unique IDs	13	54	79	106

been one of the constraints on performance over the last decade, and many studies aim to reduce the energy consumption of neural networks.

BNN can reduce power consumption for two main reasons: 1) most 32-bit arithmetic operations are replaced with bit-wise operations in BNN which has binarizing weights and activations. As fewer system resources are involved for computation, it leads to an increase in power efficiency; 2) data movement in memory is more expensive than computation. As reported from [75], in neural networks such as GoogLeNet, only 10% of the total energy is consumed due to computation. In comparison, 68% of the total energy is consumed from moving the feature maps. Compared with full-precision DNNs, the proposed BNN-based IDS can request less memory space to fit with on-chip memory, which reduces the power waste due to memory access.

VI. IMPLEMENTATION AND EVALUATION

A. DATA COLLECTION AND ANALYSIS

We use a CAN Analyzer tool - a CANalyst-II device to collect data via the OBD-II port. As an on-board diagnostic standard, the OBD-II port has been mandatory for all vehicles sold in the US since 1996. Even though it is mainly used for diagnostic and emission measurements, it can provide additional information, including engine control, body control, and chassis control information. By connecting the CAN Analyzer with the OBD-II port of a vehicle under normal operations, we collect the CAN message data without the need of any prior knowledge about the target in-vehicle network. To show that the applicability of the proposed IDS is not limited to a specific vehicle model, we conduct experiments on four datasets collected from different vehicle models of different manufacturers. All are collected when a vehicle is driven under normal operations for twenty to forty minutes. For example, the dataset of Honda Civic 2018 was collected while we were driving under normal operations for four round trips on the road segment on campus, as shown in Figure 9. Table 1 provides details of each dataset. The third row depicts the total number of messages in each dataset. The fourth row lists the number of unique message IDs which are dependent on manufacturers and vehicle models.

B. ATTACK STRATEGY

Attackers can mount attacks by injecting malicious CAN messages to the bus via an extra ECU or a compromised

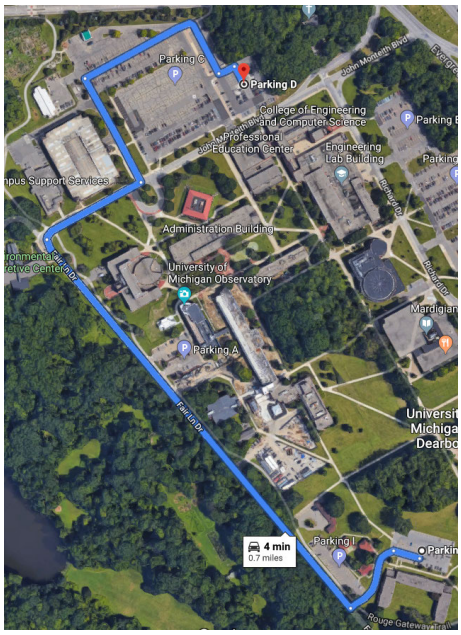

FIGURE 9. Driving path for data collection.

ECU. As shown in Figure 10, malicious messages (in red) can change the traffic sequential pattern.

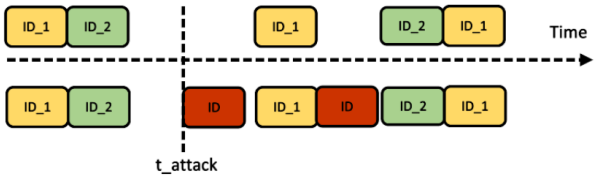

FIGURE 10. CAN traffic without and with attacks.

Considering realistic attacks in the real world and existing studies on attack strategies [2], [3], [6], [8], [13], we implement 100, 200, 200, 300 intrusions on Dataset 1, 2, 3 and 4 respectively depending on their size. Each intrusion is performed for 3 to 4 seconds. For the *all zero ID* attack, we inject a CAN message with 11 zero bits ID every 0.3 ms. The purpose of this attack is to make the network resource unavailable to intended normal functions. Since all zero ID is the most dominant CAN ID in the bus, zero ID messages always win the bus in the arbitration phase, and the DoS effect can be achieved. Consequently, all other normal CAN messages cannot be transmitted. For the *random ID* attack, we follow a similar way to the all zero ID attack with different content of malicious messages. ID and data of those malicious messages are randomized. This type of attack can result in malfunctions of the vehicle by disturbing certain functions. For both *replay* and *spoofing* attacks, we inject malicious messages every 0.8 ms. Malicious messages of the replay attack are valid messages transmitted previously on the bus. And malicious messages of the spoofing attack are altered messages based on valid messages with valid IDs and modified data. We need to understand CAN messages and know the message format to generate those malicious messages. We need to do reverse

engineering as we have no access to DBC files which are proprietary to car manufacturers. For example, we find that CAN messages with ID 0x0C8 control the instrument cluster speedometer in Dataset 1. As shown in red in Figure 11, for ID 0x0C8, only the fifth and sixth bytes in the data section are changed under normal operations. By modifying the fifth and sixth bytes, the malicious messages can mount the spoofing attack. For example, a spoofed CAN message can be generated with ID: 0x0C8 and Data: 0x00 0x00 0x00 0x00 0x01 0x02 0x00 0x00. For the hybrid attack, we inject a malicious message every 0.3 ms randomly picked from all four types of attacks.

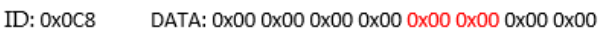

FIGURE 11. Example message for spoofing attack.

If the attacker's goal is to intentionally control the vehicle, the attacker should design malicious messages with specific ID and data to the bus. Generally, all malicious messages and normal messages can be listened by the target victim ECU. In order to ensure that malicious messages, instead of normal messages, can be received, the attacker needs to send malicious messages at higher frequency than normal messages. In a typical message injection attack, the number of attack messages should be twice the number of normal messages. In practice, the number of attack messages can be significant (20-100 times) higher than the normal messages [2].

C. EXPERIMENTAL DESIGN AND SETUP

We design and conduct experiments on different platforms. To evaluate effectiveness and efficiency of the BNN-based IDS, we use a regular desktop CPU to demonstrate the performance of our IDS in terms of accuracy and inference time on software. We use Python version 3 for the implementation and Theano/TensorFlow as libraries for machine learning. The BNN training device is a Linux machine running Ubuntu 20.04.2 LTS with a GPU device. For the hardware-based acceleration evaluation, we focus on embedded computing which aims to simulate a more realistic environment. We use two different platforms including an embedded CPU and an embedded FPGA which are low-power devices. Additionally, we use desktop CPU/GPU devices for comparison, where we compare IDS detection latency and power usage effectiveness among those devices. Details of different platforms are listed as follows:

- Embedded FPGA: Xilinx PYNQ family Artix-7 FPGA, which has 13,300 logic slices, each with four 6-input LUTs and 8 flip-flops
- Embedded CPU: Arm Cortex-A9 processor with dual-core and 650MHz clock
- Desktop CPU: AMD Ryzen 3700x with 8 cores and 16 threads, base clock at 3.6GHz
- Desktop GPU: Nvidia RTX 2070 Super, 8GB DDR6 RAM

D. INPUT GENERATOR

Our input generator assembles multiple consecutive CAN messages as input frames. Each input frame consists of N messages and padding bits. It allows the BNN model to leverage sequential patterns of the CAN traffic. We conduct experiments to show its necessity and analyze the appropriate size N of the input frame. The BNN model performs ineffectively when the IDS directly takes CAN messages without the input generator. For example, without the input generator, it achieves an accuracy of 78.43% under the hybrid attack with Dataset 1. To find the appropriate size N, we examine various frame sizes, such as 5, 10, 20, 50. Results indicate that the IDS dose not perform well when N is small because those frames may not be able to reflect sequential patterns properly. While performance is better with larger input frames that contain more messages, the performance does not consistently improve with the frame size. Based on those experiments, we design the input generator that produces frames with ten consecutive CAN messages.

E. OVERFITTING

In order to prevent overfitting, we apply the following approaches.

1) hold-out. For each dataset, we use 70% for training and 30% for testing. Considering that a CAN IDS receives CAN messages in a continuous manner and does not randomly access messages from the traffic trace, the first part of CAN traffic data is used to train our IDS while the last part data is used to test. It is noted that no cross-validation is applied because CAN IDSs cannot access messages at random from the traffic.

2) regularization. Using regularization during training can mitigate overfitting. Regularization consists of various methods. We use two methods: batch normalization and drop out layer. Batch normalization helps to prevent overfitting by reducing the internal covariate shift and instability in distributions of layer activations. Moreover, it also results in the acceleration of the model optimization and better overall performance. Drop out layers prevent overfitting by randomly dropping predefined ratio of nodes in the network.

3) characteristics of BNNs. Adding noise to activations and weights during training is one of methods to prevent overfitting. In BNNs, both the activations and the weights are binarized, which can be consider a variant of dropout method [66]. It helps to better generalize the model and reduce overfitting.

4) early stopping. We train our model for a large number of epochs, and we measure how well each iteration of the model performs. We stop the training and save the current model when it starts to degrade, which helps to prevent overfitting.

By using those approaches, we do not find overfitted models in experiments on four datasets, but it may happen for CAN data of other vehicles. As this BNN-based IDS is a scalable framework, more approaches can be applied to prevent overfitting, such as feature selection. If neural networks learn too many features, they can eventually overfit.

Feature selection helps to select important features to reduce overfitting.

F. EXPERIMENTAL RESULTS

1) BNN VS. FULL-PRECISION NN ON CPU

This experiment is to compare the inference time of BNN and full-precision 32-bit NN with the same structure. To evaluate the effectiveness under different types of attacks, we take into account existing studies and some state-of-the-art work in the CAN IDS research field [3], [46], [59]. We use three metrics: **accuracy**, **true positive rate** (TPR, or detection rate), and **false positive rate** (FPR). Accuracy is the fraction of all correct detection results. TPR is the fraction of detected messages that are truly malicious, and FPR is the fraction of detected messages that are not really malicious. Accuracy should be high as it indicates the overall detection ability, and TPR should be high as it shows the ability to detect malicious messages. FPR should be low as it means the low false alarm. To calculate them, the following terms are defined:

- *TP (true positive)*: the number of malicious messages that are correctly detected.
- *FP (false positive)*: the number of malicious messages that are incorrectly detected.
- *TN (true negative)*: the number of normal messages that are correctly detected.
- *FN (false negative)*: the number of normal messages that are incorrectly detected.

Accuracy, TPR and FPR are calculated as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{7}$$

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{9}$$

Note that high detection rate (TPR) and low false alarm rate (FPR) are considered as major performance criteria for IDSs generally. Low false positives are critical for in-vehicle networks [3], especially since false positives can lead to unnecessary panic and waste of resources, such as emergency services.

By following the common practice, for each dataset, we use 70% for training and 30% for testing. Based on our experiments, we notice that results of each dataset are comparable, thus we report the average of their performance in Table 2. It also demonstrates that the applicability of the our IDS is not limited to a specific vehicle model. Results in Table 2 are calculated over 10 experiments. It can be seen that BNN inference time is about 3 times faster than regular NN. This is an improvement for real-time applications that are time-sensitive. Authors of [14] report that their DNN-based IDS has a processing latency of 2-5 ms per message, which meets real-time requirements of CAN. Our scheme achieves shorter processing time.

BNN has a noticeable higher false positive rate compared to the 32-bit NN. For true positive rate, BNN outperforms against the all zero attack and the random ID attack in comparison of other types of attacks. As for accuracy, BNN can handle the random ID attack better than other attack types in general, but there is still room to improve when comparing with full-precision NN in terms of any attack types. The overall performance of BNN on the spoofing attack and the hybrid attack is not comparable with others, and it might be because those attacks are more complicated than other types of attacks. The proposed BNN-based IDS runs 3 times faster in detection while trading off 2.1%-18.5% accuracy depending on different attack types compared with the 32-bit NN model.

Even though we focus on neural networks for higher accuracy, we also conduct experiments to compare the performance of our scheme with some classical machine learning methods, including decision tree, random forest and SVM. These classical machine learning methods generally need more human interaction and additional efforts, such as data preprocessing and feature engineering, compared with neural networks. It has been shown that neural networks outperform classical machine learning methods, in both general IDS tasks [15], [50], [51], [52] and CAN bus IDS tasks [13], [14]. We draw the same conclusion through our experiments.

TABLE 2. BNN-based and NN-based IDSs on CPU.

		BNN-1 bit	NN-32 bits
	Inference time /message	0.04 ms	0.12 ms
All Zero ID Attack	FPR	3.75 %	0.08 %
	TPR	97.31 %	99.72 %
	Accuracy	96.53 %	99.82 %
Random ID Attack	FPR	3.32 %	0.12 %
	TPR	98.16 %	99.62 %
	Accuracy	97.84 %	99.92 %
Replay Attack	FPR	8.02 %	0.32 %
	TPR	94.32 %	99.25 %
	Accuracy	93.11 %	99.59 %
Spoofing Attack	FPR	11.03 %	0.88 %
	TPR	88.90 %	99.01 %
	Accuracy	85.12 %	98.89 %
Hybrid Attack	FPR	18.02 %	1.02 %
	TPR	80.23 %	98.02 %
	Accuracy	79.99 %	98.45 %

2) BNN IMPLEMENTATION ON FPGA, CPU AND GPU

In this experiment, we compare the performance of BNN-based IDS with different platforms. The same BNN model under the hybrid attack is used on four common hardware platforms: desktop CPU, embedded CPU (eCPU), FPGA, and GPU. We measure the power consumption of the FPGA hardware by connecting a USB power monitor to measure the total power consumption of the board and estimate the power consumption of others based on the hardware specifications, which is accurate enough to use for our purpose [27]. It is noted that the processing delay of input generator is not included. From the results, not surprising, the GPU device are the fastest inference device during our test. Inference on the FPGA device is 2.7 times faster than the CPU and 128 times faster the eCPU, because of its optimization on the bitwise operation and highly efficient parallel data

processing. When we consider the power efficiency (message/sec/watt), FPGAs are the best hardware solution in our experiment, which is 6.3 times than the GPU, 55.8 times than the CPU, and 91.6 times than the eCPU. In practical usage, using powerful GPUs or desktop CPUs is not feasible solution for ECU setup, therefore, compared with the eCPU platform, which is limited on computation resources, the FPGA platform becomes a recommended solution as a BNN-based IDS inference device.

TABLE 3. BNN-based IDS on different platforms.

	FPGA	CPU	eCPU	GPU
Power consumption (watt)	3.1	65	2.2	215
Throughput (message/sec)	55666	20950	434	613002
Efficiency (message/sec/watt)	17957	322	197	2851

3) BNN MODELS WITH WIDER/DEEPER STRUCTURE

To further improve the accuracy performance of the proposed IDS, we explore several BNN model choices with different width and depth settings to understand if and how more complicated network structures can provide better detection accuracy. We evaluate those models from the perspective of effectiveness and efficiency. We also analyze the model size of each model.

a: EFFECTIVENESS EVALUATION

Table 4 provides the accuracy performance of different model choices. We evaluate these models on the same CPU platform with the same type of attack (hybrid attack) using three metrics - FPR, TPR and accuracy. The baseline model (3 hidden layers, 1024 neurons in each hidden layer) is the model evaluated in Section VI-F1. Model 1 is a deeper model (5 hidden layers, 1024 neurons in each hidden layer), and Model 2 is a wider model (3 hidden layers, 2048 neurons in each hidden layers). As for Model 3, it is a deeper & wider BNN model (5 layers, 2048 neurons in each hidden layers). We examine all four models using binary and 32-bit precision options to provide comparisons between binary and full-precision models. We also discuss the model size of these models.

Model 3 provides the most significant accuracy improvement on both binary and 32-bit precision networks compared to other models. However, the cost for Model 3 is also apparent. For binary networks, Model 3 improves 5.67% accuracy performance over the baseline model, while the model size increases from 515 Kb to 2.79 Mb, as seen in Table 6.

Increasing width or depth independently can also improve IDS accuracy performance. For binary networks, Model 1 and Model 2 achieve 1.34% and 4.33% accuracy boost respectively, over the baseline model. As a trade-off, these two models' size increases to 787 Kb and 1.76 Mb, respectively.

We also evaluate the effectiveness of each model design choice when the network precision is 32-bit. Since the baseline model has already reached 98.45% accuracy, increasing the width or depth of the model cannot significantly improve accuracy performance. On the other hand, the model size can be expanding multiple times in pace with width and depth adding. Even for the simplest 32-bit model, the baseline model size becomes 15.13 Mb, 29.4 times larger than the binary baseline model size, and 5.4 times larger than the size of the binary Model 3 which is the best performing binary model. This experiment shows that BNN models have an advantage on the model size reduction and demonstrates how we can improve accuracy performance over the baseline model. For different applications, different model can be selected based on resources they have to meet their needs.

TABLE 4. Effectiveness evaluation - hybrid attack.

		baseline model	Model 1	Model 2	Model 3
binary	FPR	18.02%	17.55%	15.20%	12.34%
	TPR	80.23%	81.23%	84.33%	85.42%
	Accuracy	79.99%	81.33%	84.32%	85.66%
32-bit	FPR	1.02%	0.98%	0.94%	0.93%
	TPR	98.02%	98.34%	98.83%	98.88%
	Accuracy	98.45%	98.57%	98.93%	99.12%

b: EFFICIENCY EVALUATION

We evaluate efficiency performance of the baseline BNN model, Model 1 (deeper), Model 2 (wider) and Model 3 (deeper & wider) on the same CPU platform under the same type of attack.

As shown in Table 5, we measure the inference time/message for four model design choices. For binary models, the same as the effectiveness evaluation experiment, whether it is 1-bit or 32-bit precision, Model 3 increases the inference time of the baseline model the most. With the model complexity increasing, the inference time required for each message increases. This increasing inference time needs our attention because real-time embedded systems are usually susceptible to such delays.

The inference time of each binary model is about three times faster than that of the corresponding 32-bit model. This experiment suggests using BNN can effectively decrease processing delay.

TABLE 5. Latency evaluation (Inference time/message).

	baseline model	Model 1	Model 2	Model 3
binary	0.04 ms	0.08 ms	0.13 ms	0.22 ms
32-bit	0.12 ms	0.24 ms	0.41 ms	0.67 ms

TABLE 6. Model size of different models.

	Binary baseline	Binary Model 1	Binary Model 2	Binary Model 3	32-bit baseline
Size	515 KB	787 KB	1.76 MB	2.79 MB	15.13 MB

VII. DISCUSSION

As shown in Section VI, the proposed IDS can reduce the detection latency while maintaining acceptable detection rates, and the accuracy can be further improved. It is reasonable to expect that the proposed IDS can be available for real-time detection. In this section, we list some remaining open questions, challenges and limitations:

Root cause analysis: Our IDS can respond by raising an alarm if an input is identified as malicious. However, we do not provide a root cause analysis report for further troubleshooting. The root cause can be facilitated by a fingerprinting approach [3] that can be complimentary with our IDS. As our work focuses on intrusion detection, root cause analysis is beyond the scope of this research.

Data collection: We collect CAN bus data through the OBD-II port. However, there are some practical limitations. Firstly, a car manufacturer may define some discretionary pins in the OBD-II port in addition to standard CAN high and CAN low pins if a vehicle has multiple CAN buses. Information about these pins is required in order to collect data on all CAN buses. Secondly, data collected directly from the OBD-II port may be limited or hindered because of different CAN implementations by different vehicle manufacturers. Those potential issues can be solved by connecting directly to the CAN bus but require some levels of destruction to the vehicle.

Various driving conditions: We collect datasets on the clear driving roads with cautious driving. Road conditions and driving styles can be very different in reality. These may lead to some changes in characteristics of CAN messages or even cause false alarms to an IDS. For example, driving on a slippery road or unique driving styles or habits of different drivers may cause changes in the CAN traffic. In our future work, we consider collecting data on various driving conditions to understand whether and how they affect the results.

Training model updates: Updates on the training model for detection may be needed during the life cycle of a vehicle. It can be a problem for all machine learning-based IDSs. For example, replacement or maintenance of some components may require a newly trained model. The aging of the vehicle is another concern that may require a refined trained model. The trained model can be generated offline by an authorized dealership. However, the updating process may expose the system to other threats or attacks.

Adversarial evasion: If an attacker can modify ECU(s) and cause a ''modified'' CAN trace for the offline training, the attacker can trick the IDS and lead to a ''wrong'' model. Then, malicious behaviors may bypass the detection of IDS. Our assumption is that the CAN messages collected for training are the normal CAN traffic without any adversary's manipulations.

Limitations: The proposed BNN-based IDS demonstrate fast detection and satisfying detection performance especially for certain types of attacks as shown in experimental results. There is a trade-off between detection performance and processing delay. Both true detection rates and accuracy are lower while false positive rates are still higher than full-precision one for all types of attacks. To improve performance, using complex neural network structure or leveraging features derived from the semantic meaning of CAN traffic can be a solution. In addition, there is a limitation that all supervised learning models have, including neural networks that are used in supervised learning for IDSs, such as ANN, RNN and CNN [13], [14], [15]. They can be limited to detecting unlearned types of attacks, and additional research in this direction is needed.

VIII. CONCLUSION

This paper proposes a BNN-based IDS to secure the in-vehicle CAN bus and protect vehicles from malicious intrusions. The proposed IDS utilizes BNN to accelerate intrusion detection while reducing detection latency, memory request and system power consumption. The IDS, which is designed to suit CAN data, can process the raw CAN data without additional prepossessing or hand-designed features. It learns from sequential patterns from CAN traffic rather than individual CAN message. We evaluate the proposed IDS on datasets collected from four real vehicles of different models and manufacturers, which also helps validate that our IDS's applicability is not limited to a specific vehicle model. Experimental results demonstrate that the IDS is able to detect malicious behaviors with lower latency (3 times faster) compared to full-precision neural network-based IDS with sacrificing the accuracy (2.1%-18.5% depending on the types of attacks) on the same CPU platform. And the model runs 128 times faster on FPGAs than an embedded CPU. Moreover, the experiments show a BNN model with a wider or deeper structure can achieve better accuracy, but as a trade-off, it increases detection latency and model size to varying degrees. In addition, experimental results present that the BNN model provides a significant advantage on the model size reduction compared to 32-bit counterparts. Therefore, different applications can select the proper model according to their needs and resources.

Our study can serve as a proof-of-concept that our BNN-based IDS has a potential to enhance in-vehicle security significantly, especially considering the embedded environments of in-vehicle networks. In the future, we will explore other improvement strategies that can improve performance while maintaining its low latency advantages. From experiment results, the hybrid attack is more difficult to be detected than others. One of our focuses will be to reveal how the hybrid attack leads to performance degradation, and then design an IDS that can better defend against such attack. We also plan to conduct research to consider other CAN data features, such as semantic features by collaborating with manufacturers, to improve the BNN model against sophisticated attacks. Another direction of our future work is to incorporate more attack types to evaluate the effectiveness of our IDS, such as unknown attacks generated by adversarial training techniques.

REFERENCES

[1] G. Xie, L. T. Yang, Y. Yang, H. Luo, R. Li, and M. Alazab, "Threat analysis for automotive CAN networks: A GAN model-based intrusion detection technique," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4467–4477, Jul. 2021.

[2] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," *Black Hat USA*, vol. 2014, p. 94, Aug. 2014.

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

[4] S. Nie, L. Liu, Y. Du, and W. Zhang, "Over-the-air: How we remotely compromised the gateway, BCM, and autopilot ECUs of Tesla cars," *Briefing, Black Hat USA*, pp. 1–19, Aug. 2018.

[5] M. Pham and K. Xiong, "A survey on security attacks and defense techniques for connected and autonomous vehicles," *Comput. Secur.*, vol. 109, Oct. 2021, Art. no. 102269.

[6] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S91, pp. 1–92, 2015.

[7] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy*. Washington, DC, USA: IEEE Computer Society, May 2010, pp. 447–462.

[8] C. Miller and C. Valasek, "Adventures in automotive networks and control units," in *Proc. DEF CON*, Aug. 2013, pp. 15–31.

[9] Y. D. Sen Nie and L. Liu, "Free-fall: Hacking Tesla from wireless to can bus," *Black Hat USA*, vol. 25, pp. 1–16, Jul. 2017.

[10] L. Wouters, B. Gierlichs, and B. Preneel, "My other car is your car: Compromising the Tesla model X keyless entry system," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, no. 4, pp. 149–172, Aug. 2021.

[11] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. 20th USENIX Conf. Secur.* Berkeley, CA, USA: USENIX Association, 2011, p. 6.

[12] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive CAN networks-practical examples and selected short-term countermeasures," *Rel. Eng. Syst. Saf.*, vol. 96, no. 1, pp. 11–25, 2011.

[13] 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.

[14] 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, Jun. 2016, Art. no. e0155781.

[15] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2017, pp. 313–316.

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[17] Y. B. Ian Goodfellow and A. Courville, *Deep Learning Book*. Cambridge, MA, USA: MIT Press, 2016.

[18] K. Siu, D. M. Stuart, M. Mahmoud, and A. Moshovos, "Memory requirements for convolutional neural network hardware accelerators," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Sep. 2018, pp. 111–121.

[19] S. Zhang, H. Jiang, S. Xiong, S. Wei, and L.-R. Dai, "Compact feedforward sequential memory networks for large vocabulary continuous speech recognition," in *Proc. Interspeech*, 2016, pp. 3389–3393.

[20] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64270–64277, 2018.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[22] X. Sun, S. Yin, X. Peng, R. Liu, J.-S. Seo, and S. Yu, "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1423–1428.

[23] K. Abdelouahab, M. Pelcat, J. Serot, and F. Berry, "Accelerating CNN inference on FPGAs: A survey," 2018, *arXiv:1806.01683*.

[24] S. I. Venieris, A. Kouris, and C.-S. Bouganis, "Toolflows for mapping convolutional neural networks on FPGAs: A survey and future directions," 2018, *arXiv:1803.05900*.

[25] Z. Li, Y. Zhang, J. Wang, and J. Lai, "A survey of FPGA design for AI era," *J. Semiconductors*, vol. 41, no. 2, Feb. 2020, Art. no. 021402.

[26] D. J. M. Moss, E. Nurvitadhi, J. Sim, A. Mishra, D. Marr, S. Subhaschandra, and P. H. W. Leong, "High performance binary neural networks on the Xeon+FPGA platform," in *Proc. 27th Int. Conf. Field Program. Log. Appl. (FPL)*, Sep. 2017, pp. 1–4.

[27] R. Zhao, W. Song, W. Zhang, T. Xing, J.-H. Lin, M. Srivastava, R. Gupta, and Z. Zhang, "Accelerating binarized convolutional neural networks with software-programmable FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2017, pp. 15–24.

[28] A. Prost-Boucle, A. Bourge, F. Petrot, H. Alemdar, N. Caldwell, and V. Leroy, "Scalable high-performance architecture for convolutional ternary neural networks on FPGA," in *Proc. 27th Int. Conf. Field Program. Log. Appl. (FPL)*, Sep. 2017, pp. 1–7.

[29] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A survey of FPGA-based neural network accelerator," 2017, *arXiv:1712.08934*.

[30] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "FINN: A framework for fast, scalable binarized neural network inference," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2017, pp. 65–74.

[31] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 525–542.

[32] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–9.

[33] A. Bulat and G. Tzimiropoulos, "Binarized convolutional landmark localizers for human pose estimation and face alignment with limited resources," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3706–3714.

[34] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2017.

[35] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'brien, Y. Umuroglu, M. Leeser, and K. Vissers, "FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 3, pp. 1–23, 2018.

[36] L. Zhang, X. Yan, and D. Ma, "Accelerating in-vehicle network intrusion detection system using binarized neural network," SAE Tech. Paper 2022-01-0156, 2022.

[37] CAN Specification, "Bosch," Robert Bosch GmbH, Postfach, 1991, vol. 50. [Online]. Available: http://lbenchindia.cfavourites.com/finalyearprojectdatasheets/can2spec.pdf

[38] M. Di Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice*. Springer, 2012. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4614-0314-2

[39] S. Corrigan, "Introduction to the controller area network (CAN)," HPL, Texas Instrum., Dallas, TX, USA, Appl. Rep. SLOA101, 2002, pp. 1–17. [Online]. Available: https://www.rpi.edu/dept/ecse/mps/sloa101.pdf

[40] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Syst.*, vol. 35, no. 3, pp. 239–272, Apr. 2007.

[41] J. Liu, S. Zhang, W. Sun, and Y. Shi, "In-vehicle network attacks and countermeasures: Challenges and future directions," *IEEE Netw.*, vol. 31, no. 5, pp. 50–58, May 2017.

[42] K. Kim, J. S. Kim, S. Jeong, J.-H. Park, and H. K. Kim, "Cybersecurity for autonomous vehicles: Review of attacks and defense," *Comput. Secur.*, vol. 103, Apr. 2021, Art. no. 102150.

[43] M. Muter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 1110–1115.

[44] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *Proc. Int. Conf. IEEE Inf. Netw. (ICOIN)*, Jan. 2016, pp. 63–68.

[45] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni, "Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms," in *Proc. IEEE 2nd Int. Forum Res. Technol. Soc. Ind. Leveraging*, Oct. 2016, pp. 1–6.

[46] Q. Wang, Z. Lu, and G. Qu, "An entropy analysis based intrusion detection system for controller area network in vehicles," in *Proc. 31st IEEE Int. Syst. Chip Conf. (SOCC)*, Sep. 2018, pp. 90–95.

[47] S. Ohira, A. K. Desta, I. Arai, and K. Fujikawa, "PLI-TDC: Super fine delay-time based physical-layer identification with time-to-digital converter for in-vehicle networks," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, May 2021, pp. 176–186.

[48] S. C. Kalkan and O. K. Sahingoz, "In-vehicle intrusion detection system on controller area network with machine learning models," in *Proc. 11th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2020, pp. 1–6.
[49] F. Nappi, "A survey of intrusion detection systems for controller area networks and FPGA evaluation," M.S. thesis, 2022. [Online]. Available: https://www.politesi.polimi.it/bitstream/10589/186244/1/Thesis.pdf
[50] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
[51] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[52] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: A survey and an objective comparison," *J. Netw. Comput. Appl.*, vol. 169, Nov. 2020, Art. no. 102767.
[53] S. N. Narayanan, S. Mittal, and A. Joshi, "Obd_SecureAlert: An anomaly detection system for vehicles," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, May 2016, pp. 1–6.
[54] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," in *Proc. World Congr. Ind. Control Syst. Secur. (WCICSS)*, Dec. 2015, pp. 45–49.
[55] A. Wasicek, M. D. Pesé, A. Weimerskirch, Y. Burakova, and K. Singh, "Context-aware intrusion detection in automotive control systems," in *Proc. 5th ESCAR*, Ypsilanti, MI, USA, 2017, pp. 21–22.
[56] Z. Wei, Y. Yang, Y. Rehana, Y. Wu, J. Weng, and R. H. Deng, "IoVShield: An efficient vehicular intrusion detection system for self-driving," in *Proc. Int. Conf. Inf. Secur. Pract. Exper.*, Cham, Switzerland: Springer, 2017, pp. 638–647.
[57] 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, Secur. Trust (PST)*, Aug. 2018, pp. 1–6.
[58] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "Long short-term memory-based intrusion detection system for in-vehicle controller area network bus," in *Proc. IEEE 44th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2020, pp. 10–17.
[59] L. Zhang and D. Ma, "A hybrid approach toward efficient and accurate intrusion detection for in-vehicle networks," *IEEE Access*, vol. 10, pp. 10852–10866, 2022.
[60] L. Zhang, L. Shi, N. Kaja, and D. Ma, "A two-stage deep learning approach for can intrusion detection," in *Proc. Ground Vehicle Syst. Eng. Technol. Symp. (GVSETS)*, 2018, pp. 1–11.
[61] N. Oliveira, I. Praça, E. Maia, and O. Sousa, "Intelligent cyber attack detection and classification for network-based intrusion detection systems," *Appl. Sci.*, vol. 11, no. 4, p. 1674, Feb. 2021.
[62] Z. Xing, J. Pei, and E. Keogh, "A brief survey on sequence classification," *ACM SIGKDD Explor. Newslett.*, vol. 12, no. 1, pp. 40–48, Jun. 2010.
[63] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, 2017, pp. 712–717.
[64] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.
[65] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," 2016, *arXiv:1602.07360*.
[66] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1," 2016, *arXiv:1602.02830*.
[67] W. Tang, G. Hua, and L. Wang, "How to train a compact binary neural network with high accuracy?" in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1–7.
[68] A. Putnam, "FPGAs in the datacenter: Combining the worlds of hardware and software development," in *Proc. Great Lakes Symp. VLSI*, May 2017, p. 5.
[69] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2015, pp. 161–170.
[70] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-Eye: A complete design flow for mapping CNN onto embedded FPGA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 35–47, Jan. 2017.
[71] S. Liang, S. Yin, L. Liu, W. Luk, and S. Wei, "FP-BNN: Binarized neural network on FPGA," *Neurocomputing*, vol. 275, pp. 1072–1086, Jan. 2018.
[72] M. Marchetti and D. Stabili, "Anomaly detection of CAN bus messages through analysis of ID sequences," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 1577–1583.
[73] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary neural networks: A survey," *Pattern Recognit.*, vol. 105, Sep. 2020, Art. no. 107281.
[74] C. Hao, X. Zhang, Y. Li, S. Huang, J. Xiong, K. Rupnow, W.-M. Hwu, and D. Chen, "FPGA/DNN co-design: An efficient design methodology for IoT intelligence on the edge," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.
[75] T.-J. Yang, Y.-H. Chen, J. Emer, and V. Sze, "A method to estimate the energy consumption of deep neural networks," in *Proc. 51st Asilomar Conf. Signals, Syst., Comput.*, Oct. 2017, pp. 1916–1920.

LINXI ZHANG (Member, IEEE) received the B.S. degree from the Department of Computer Science and Engineering, University of Electronic Science and Technology of China, in 2014, and the M.S. degree from the Department of Computer and Information Science, University of Michigan–Dearborn, USA, where she is currently pursuing the Ph.D. degree in computer and information. Her research interests include automotive security, in-vehicle network security, machine learning, and smartphone security.

XUKE YAN (Member, IEEE) received the B.S. degree from the Department of Electrical Engineering, University of Electronic Science and Technology of China, in 2014, and the M.S. degree from the Department of Electrical and Computer Engineering, University of Michigan-Dearborn, in 2015. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Oakland University, USA. His research interests include machine learning, computer vision, and embedded systems.

DI MA (Senior Member, IEEE) received the Ph.D. degree from the University of California, Irvine, in 2009. She was at the IBM Almaden Research Center in 2008 and the Institute for Infocomm Research, Singapore in from 2000 to2005. She is currently a Professor with the Computer and Information Science (CIS) Department, College of Engineering and Computer Science (CECS), University of Michigan–Dearborn. She is also serving as the Interim Associate Dean for the Graduate Education and Research and the Director of the Cybersecurity Center for Education, Research, and Outreach, CECS. She is broadly interested in the general area of security, privacy, and applied cryptography. Her research was supported by NSF, NHTSA, AFOSR, Intel, Ford, and Research in Motion. Her research interests include connected and autonomous vehicle security, smartphone and mobile device security, RFID and sensor security, and data privacy. She was a recipient of the Trevor O. Jones Outstanding Paper Award from Society of Automobile Engineers (SAE) in 2019, the Distinguished Research Award from the College of Engineering and Computer Science of UM-Dearborn in 2017, and the Tan Kah Kee Young Inventor Award in 2004.