Data-Driven Intrusion Detection for Intelligent Internet of Vehicles: A Deep Convolutional Neural Network-based Method

Laisen Nie, Zhaolong Ning, Xiaojie Wang, Xiping Hu, Yongkang Li, Jun Cheng

***Abstract*—As an industrial application of Internet of Things (IoT), Internet of Vehicles (IoV) is one of the most crucial techniques for Intelligent Transportation System (ITS), which is a basic element of smart cities. The primary issue for the deployment of ITS based on IoV is the security for both users and infrastructures. The Intrusion Detection System (IDS) is important for IoV users to keep them away from various attacks via the malware and ensure the security of users and infrastructures. In this paper, we design a data-driven IDS by analyzing the link load behaviors of the Road Side Unit (RSU) in the IoV against various attacks leading to the irregular fluctuations of traffic flows. A deep learning architecture based on the Convolutional Neural Network (CNN) is designed to extract the features of link loads, and detect the intrusion aiming at RSUs. The proposed architecture is composed of a traditional CNN and a fundamental error term in view of the convergence of the backpropagation algorithm. Meanwhile, a theoretical analysis of the convergence is provided by the probabilistic representation for the proposed CNN-based deep architecture. We finally evaluate the accuracy of our method by way of implementing it over the testbed.**

***Index Terms*—Internet of vehicles, intrusion detection, data-driven, convolutional neural network, smart cities.**

I. Introduction

With the rapid development of Internet of Things (IoT), it has been extended to vehicles for structuring the Intelligent Transportation System (ITS), i.e., the intelligent Internet of Vehicles (IoV). The intelligent IoV can provide credible and efficient communication services for vehicles to support kinds of applications, e.g., the Global Positioning System (GPS) and the advanced driving assistance system [1]. It is one of the most crucial techniques for employing the ITS in order to realize smart cities. Recently, various networking techniques, such as the 5G-enable communication network, have been already involved in IoVs to provide the higher bandwidth for users. Subsequently, an increasing number of users' private information is transmitted over IoVs. Furthermore, the IoV is an open network for each user, though the user may be a possible attacker. Due to the openness of the IoV, it is usually attacked by various malware (e.g., the worm and the Trojan horse). Meanwhile, the Distributed Denial of Service (DDoS) attack has been a main menace for vehicles, in which the flooding DDoS attack known as a brute-force attack exhausts the cache and computing resources of vehicles. Thereby, to protect the privacy and security of users, the intrusion detection techniques have obtained more and more attention in IoVs [2]–[7].

Currently, the taxonomy of data-driven intrusion detection methods consists of the signature-based methods and the anomaly-based methods [8]–[11]. The signature-based methods take advantage of a database in an Intrusion Detection System (IDS), reporting the abnormal pattern signatures of known intrusions to implement intrusion detection. In this case, the IDS matches the collected features of attacks with the signatures to decide whether an intrusion occurs or not. The main issue of the signature-based methods is the weakness for the intrusions from the novel and unknown attacks. The anomaly-based methods analyze the network data (such as network traffic), and detect attacks according to the abnormal behaviors of network data. Though the anomaly-based methods are able to detect unknown attacks from novel malware, they suffer from the high false alarm rate.

The approaches based on machine learning are ubiquitous for dealing with the issues of the anomaly-based methods. However, considering the specific scenario of an IoV, there are several challenges to be solved which can be summarized as follows [12]–[16]:

- The nodes in an IoV may be limited in computing and memory [17]. In this case, it is rather difficult to deploy the network data sampling function on each node of IoVs for implementing a data-driven IDS. That is because both data sampling and analyzing will consume a great number node resources. Therefore, running a data-driven IDS has a negative effect on nodes, which may reduce the throughput of an IoV.
- Vehicles usually move fast, which means that the topology of an IoV changes frequently and nodes access to the network randomly [18]. Moreover, the links of an IoV are always short-time. Nevertheless, the machine learning-based approaches often need a lot of network data to identify anomalies for intrusion detection.
- Most of previous approaches based on machine learning analyze anomalous network data collecting by the On-

L. Nie and Y. Li are with the School of Electronics and Information, Northwestern Polytechnical University, Xi'an, 710072, China, and with Qingdao Research Institute of Northwestern Polytechnical University, Qingdao, China (email: nielaisen@nwpu.edu.cn, ykli@nwpu.edu.cn).

Z. Ning (Co-coresponding author) is with the School of Software, Dalian University of Technology, Dalian 116024, China, (e-mail: zhaolongning@dlut.edu.cn).

X. Wang (Co-Corresponding author) is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China (email: xiaojie.wang@polyu.edu.hk).

X. Hu (Co-coresponding author) and J. Cheng (Co-coresponding author) are with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: xp.hu@siat.ac.cn, jun.cheng@siat.ac.cn).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.2990984, IEEE Transactions on Network Science and Engineering

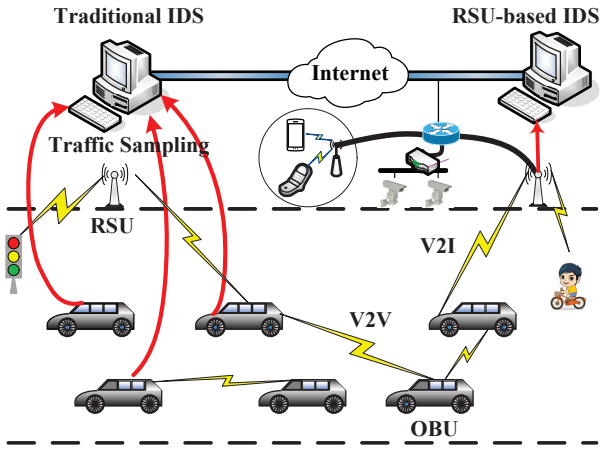

Fig. 1. Illustration of IDS based on RSU.

Board Unit (OBU). The complicated network topology as well as limited computing and cache capacity in IoVs impede the network managers to sample adequate network data using for the anomaly-based intrusion detection.

Motivated by these observations, we focus on the problem of intrusion detection in an IoV, and propose a data-driven and anomaly-based detection approach by analyzing the network traffic data from the egress points (i.e., RSUs), as shown in Fig. 1. In our method, we assume that the coverage areas of all RSUs are not overlapped. Then, each RSU identifies the intrusions of the corresponding OBUs independently. Namely, the RSU detects the intrusions of the OBUs accessing to this RSU. For convenience, we just present the intrusion detection mechanism in an RSU as an example throughout this paper. In details, we first describe the link loads entering the network by a matrix, and then design a deep architecture based on the Convolutional Neural Network (CNN) with a redundant error term to extract the features of link loads. The CNN is a prevalent technique to extract the spatio-temporal feature of a two-dimensional dataset. In image processing, pattern recognition and computer vision, the CNN is a powerful tool to model the dependence of two-dimensional dataset. By contrast, the matrix that shows the volume of traffic also yields a spatio-temporal feature. Hence, in our work, we use the CNN to model the statistical features of link loads. In fact, the intrusion detection can be formulated by the hypothesis testing. Namely, the destination of the designed deep architecture is for classification. As a result, a fully connected layer and an output layer with two neurons are used to carry out the supervised learning for intrusion detection. In addition, we also take into account the real-time performance of the proposed deep architecture, and refer to an additional error term on the output layer. The additional error term can improve the convergence of the deep architecture.

The main contributions of this paper are summarized as follows:

- We design an effective data-driven IDS by way of analyzing the link loads of RSUs instead of the nodes in the network. Different from most of the traditional IDSs carry out the network data sampling on the OBU all the time, considering the limited resources of OBUs in an IoV, the proposed data-driven IDS collects network data from the RSU.
- We propose a deep architecture based on CNN with 7 layers for intrusion detection. The spatial feature of link loads is utilized to design the loss function. Based on the deep architecture, a redundant error term on the output layer is employed to enhance the convergence of the training error. This term derives a feedback of error from the output map, which increases the residual error during the backpropagation process so that the parameters of the deep architecture are rapidly updated without loss of rationality.
- We analyze the convergence of the training error in depth. A probabilistic representation of the deep architecture is brought in our evaluation. The deep architecture based on CNN is constructed as a Bayesian hierarchical model, and then we prove the prominent convergence of our deep architecture with a redundant error term in training error.

The remainder of this paper is organized as follows. We introduce the related work in Section II. Section III describes the data-driven IDS. Numerical results are provided to evaluate the proposed intrusion detection approach in Section IV. Section V concludes our work and summarizes the future work about data-driven intrusion detection in intelligent IoVs.

II. Related Work

Initially, the signature-based (or host-based) intrusion detection methods have been progressed widely, so that the accuracy of the IDS is guaranteed for the real-time intrusion detection in an IoV [19]–[21]. In [12], the authors used a statistical technique to analyze the traffic flows of IoV to find out the rogue nodes and detect other attacks. This method first collected network traffic flows, and then modeled the intrusion detection problem as a hypothesis testing. Based on the insight analysis of traffic flows, the designed IDS can make a decision to accept or reject data. The proposed intrusion detection mechanism had a good performance to detect both rogue nodes and other attacks. Nevertheless, when several attacks appeared simultaneously, the accuracy deceased dramatically. Aiming at the DDoS attack, integrity target, and false alert's generation, the authors in [15] designed a light-weight IDS. This IDS relied on relevant detection rule to identify the corresponding attacks, namely it detected each attack independently.

Recently, with the rapidly development of artificial intelligent, the machine learning-based intrusion detection has obtained a sufficient development [22], [23]. The authors in [24] studied the problem of intrusion detection and its overhead, and proposed an intrusion detection mechanism based on the Bayesian game model, in which the trade-off between false positive rates and overhead were taken into account. Besides, the authors in [8] used the online sequential extreme learning machine to design a distributed IDS based on fog computing in IoTs. Comparing with the traditional and centralized IDS, this IDS detected an intrusion and interpreted the attacker by way of the fog nodes. Aiming at botnets, the authors in [25]

proposed a traffic-based IDS, in which a network feature set and a feature selection technique were designed to implement decision making for botnet detection. Nowadays, the deep learning technique has been brought in intrusion detection to decrease the false alarm rate, such as the hierarchical deep architecture consisting of the recurrent neural network and the multilayer perceptron. In addition, the authors in [26] studied the IDS in software defined networks, and designed a simple feedforward neural network with 3 hidden layers for intrusion detection.

Comparing with the traditional Internet service provider network, wireless networks (e.g., wireless sensor networks and mobile ad-hoc networks) have significant differences, such as limited cache and computing resources of nodes. Thus, the emphasis of the intrusion detection approach for wireless network is obtaining a tradeoff between the false alarm rate and network resources. Motivated by this issue, the cooperative game theory was utilized in [27] to increase the efficiency of the IDS so as to obtain a tradeoff between security and energy. The authors in [28] studied the intrusion detection problem for a specific paradigm of wireless sensor networks, i.e., the visual sensor network. The proposed approach was also based on the feature extraction of network traffic. They proposed the correlated N-Burst model for feature extraction at first, and then the hierarchical self-organizing map neural network was adopted for binary classification. To decrease the memory and computational consumption of nodes, a principal component analysis-based intrusion detection method was proposed in [29], in which an improved V-detector algorithm and a multi-levels detection model were used jointly.

To the best of our knowledge, the previous intrusion detection approaches are employed within the network, namely the network data come from network inside. Sampling these datasets will consume a lot of network resources. For instance, transmitting these datasets may occupy the bandwidth of an IoV, and sampling these datasets reduces the capacity of IoV's memory. Thereby, it is still significantly difficult to gain an available IDS with lower false alarm rate and consumption.

III. Our Methodology

A. Convolutional Neural Network

A convolutional neural network is a hierarchical system made up of neurons [30]–[33]. For a single neuron, it is formulated by a function with input $\boldsymbol{X}$ and output a. The function can be denoted by:

$$a^{(l)} = f\left(\boldsymbol{X}^{(l-1)} * \boldsymbol{w}^{(l-1)} + b^{(l-1)}\right), \tag{1}$$

where $\boldsymbol{w}$ is the weight vector, and scalar b is the bias of the neuron. Function $f(\cdot)$ is known as the activation function. The activation function has various choices, e.g., the sigmoid function, the Rectified Linear Unit (ReLU), the hyperbolic tangent function and so on. The process of training is to explore a group of parameters (e.g., weights and biases), so that the neural network can fit the relationship between inputs and outputs. The prevalent training approach is the backpropagation algorithm, in which a loss function is defined at first. Generally, the loss function for a single training example $\left(\boldsymbol{X}^{(m)}, \boldsymbol{y}^{(m)}\right)$ can be described as:

$$L\left(w, b, \boldsymbol{X}^{(m)}, \boldsymbol{y}^{(m)}\right) = \frac{1}{2}\left\|h_{w,b}\left(\boldsymbol{X}^{(m)}\right) - \boldsymbol{y}^{(m)}\right\|^2, \tag{2}$$

where $h_{w,b}\left(\boldsymbol{X}^{(m)}\right)$ is the output of the neural network, and notation $\|\cdot\|$ is the ℓ_2-norm. For M training examples, the loss function is:

$$L(w, b) = \frac{1}{m}\sum_{m=1}^{M} L\left(w, b, \boldsymbol{X}^{(m)}, \boldsymbol{y}^{(m)}\right). \tag{3}$$

The backpropagation algorithm searches the optimal solution by minimizing the loss function utilizing the gradient descent algorithm. The partial derivative in the gradient descent algorithm denoted by $\partial L \Big/ \partial w_{i,j}^{(l)}$ and $\partial L \Big/ \partial b_i^{(l)}$ can be computed by the backpropagation of an error term. Herein, $w_{i,j}^{(l)}$ is the weight associated with the connection between neuron j in the l-th layer and neuron i in the $l+1$-th layer, and $b^{(l)}$ shows the bias associated with neuron i in the $l+1$-th layer. Then, the parameters are updated by:

$$\begin{cases} w_{i,j}^{(l)} = w_{i,j}^{(l)} - \alpha \frac{\partial L}{\partial w_{i,j}^{(l)}}, \\ b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial L}{b_i^{(l)}}, \end{cases} \tag{4}$$

where α is the learning rate. To obtain these partial derivatives, the error term is defined as the error between the training example $\boldsymbol{y}^{(m)}$ and the corresponding output map $h_{w,b}\left(\boldsymbol{X}^{(m)}\right)$. For any neuron of the output layer, the error term is defined as:

$$e_i^{(n_l)} = -\left(y_i - a_i^{(n_l)}\right) \times f'\left(z_i^{(n_l)}\right), \tag{5}$$

where n_l is the number of layers, and $i \in \left\{S^{(l)}\right\}$. $S^{(l)}$ denotes the number of neurons in the l-th layer. Then, the backpropagation for the convolution layers is given as:

$$e_i^{(l)} = \left(e_i^{(l+1)} \otimes \mathbf{1}_{Q\times Q}\right) \circ f'\left(z_i^{(l)}\right), \tag{6}$$

where "$\circ$" and "$\otimes$" denote the Hadamard product and the Kronecker product, respectively. Q is the factor of the following pooling layer. For a pooling layer followed by a convolution layer, it is:

$$e_i^{(l)} = \left(e_i^{(l+1)} * r\left(\mathrm{w}_{\mathrm{i}}^{(\mathrm{l}+1)}\right)\right) \circ f'\left(z_i^{(l)}\right). \tag{7}$$

Function $r(\cdot)$ denotes the rotation operation of a matrix, in which it is rotated by 180°. Then, the partial derivatives for the convolution layer are computed by:

$$\begin{cases} \frac{\partial L}{\partial w_{i,j}^{(l)}} = \sum_{u,v}\left(e_i^{(l)}\right)_{u,v}\left(a_j^{(l-1)}\right)_{u,v}, \\ \frac{\partial L}{\partial b_i^{(l)}} = \sum_{u,v}\left(e_i^{(l)}\right)_{u,v}, \end{cases} \tag{8}$$

where $\left(a_j^{(l-1)}\right)_{u,v}$ is the element at (u, v) in the output map of the $l-1$-th layer.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.2990984, IEEE Transactions on Network Science and Engineering

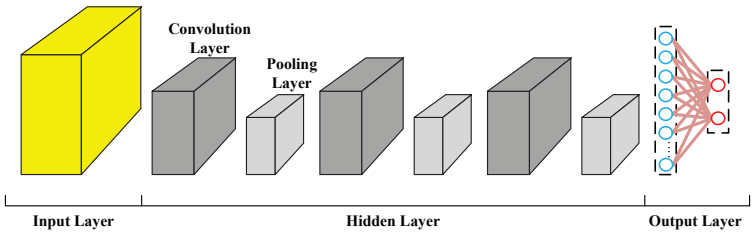

Fig. 2. CNN architecture for intrusion detection.

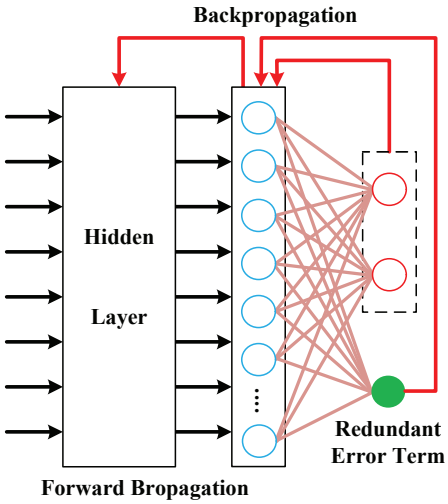

Fig. 3. Output map of the CNN architecture with redundant error term.

B. Deep Architecture based on CNN for Intrusion Detection

The deep architecture based on CNN designed for intrusion detection is made up of 6 hidden layers (3 convolution layers and 3 pooling layers), as shown in Fig. 2. Three convolution layers learn 6, 6 and 12 convolution kernels of size 5×5. The fully connected layer is used in this deep architecture, and then two single neurons are employed for classification. Three pooling layers implement the average pooling with a factor of 2. Essentially, the intrusion detection is a problem of classification, thus the sigmoid function is brought in the deep architecture.

Generally, the link load is a time series in the sense that we can denote it by a vector. Besides, the link load obeys various statistical features, just as the same as the end-to-end network traffic. Therefore, we take into account the spatial features of link load, and denote it by a matrix. We assume that the link load of an RSU is denoted by the square matrix $\boldsymbol{X}$, where $\boldsymbol{X}=[\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N]$ and $\boldsymbol{x}_n \in \boldsymbol{R}^N$. For a training example $(\boldsymbol{X}, \boldsymbol{y})$, $\boldsymbol{y}$ is the output map that shows whether the network is intruded or not. The output map of the first convolution layer can be calculated by:

$$\boldsymbol{a}_j^{(2)} = sigmoid\left(\boldsymbol{X} * \boldsymbol{K}_j^{(1)} + b_j^{(1)}\right), \quad j \in S^{(1)}, \tag{9}$$

where $sigmoid(\cdot)$ denotes the sigmoid function utilized in our deep architecture. $S^{(1)}$ represents the set of kernels in the first convolution layer. In an IoV, the traffic flows have much more irregular fluctuations comparing with other networks, such as an IP backbone network. As an aggregation of traffic flows, the link loads of RSUs also yield these fluctuations, which arises the main challenge of intrusion detection in an IoV. In order to extract irregular fluctuations and other spatial features, we use the average pooling with a factor of 2, which can be described as follows:

$$\boldsymbol{a}_j^{(3)} = average\left(\boldsymbol{a}_j^{(2)}\right), \quad j \in S^{(2)}, \tag{10}$$

where $S^{(2)}$ is a selection of input maps in the first pooling layer, and $average(\cdot)$ describes the average pooling with a factor of 2. Similarly, the forward propagation of the following convolution layers and pooling layers is given by:

$$\begin{cases} \boldsymbol{a}_j^{(4)} = sigmoid\left(\sum\limits_{i \in Z^{(2)}} \boldsymbol{a}_i^{(3)} * \boldsymbol{K}_{i,j}^{(3)} + b_j^{(3)}\right), & j \in S^{(3)}, \\ \boldsymbol{a}_j^{(5)} = average\left(\boldsymbol{a}_j^{(4)}\right), & j \in S^{(4)}, \\ \boldsymbol{a}_j^{(6)} = sigmoid\left(\sum\limits_{i \in Z^{(3)}} \boldsymbol{a}_i^{(5)} * \boldsymbol{K}_{i,j}^{(5)} + b_j^{(5)}\right), & j \in S^{(5)}, \\ \boldsymbol{a}_j^{(7)} = average\left(\boldsymbol{a}_j^{(6)}\right), & j \in S^{(6)}, \end{cases} \tag{11}$$

where $Z^{(l)}$ is the set of output maps of layer l. The intrinsic problem of intrusion detection is a classifying operation. Hence, we put forward a fully connected layer before the output layer, in which the sigmoid function is also used as the activation function. In terms of the architecture of the fully connected layer and the output layer, we have the following output map:

$$h_{w,b}(\boldsymbol{X}) = sigmoid\left(\sum_{i \in Z^{(2)}} \boldsymbol{a}_i^{(7)} * \boldsymbol{w}_{i,j}^{(7)} + b_j^{(7)}\right), \quad j \in S^{(7)}. \tag{12}$$

To train the proposed deep architecture based on CNN, the backpropagation algorithm is put forward for updating the parameters including kernels and biases. During the training process, we take into account a loss function based on ℓ_1-norm, defined as:

$$L(w, b, \boldsymbol{X}, \boldsymbol{y}) = \frac{1}{2}\left\|h_{w,b}(\boldsymbol{X}) - \boldsymbol{y}\right\|^2. \tag{13}$$

Considering M training examples, Eq. (13) is expanded as:

$$L(w, b) = \frac{1}{M}\sum_{m=1}^{M} L\left(w, b, \boldsymbol{X}^{(m)}, \boldsymbol{y}^{(m)}\right). \tag{14}$$

All of the parameters in each layer are initialized via a uniform distribution with zero-mean.

C. Deep Architecture with Redundant Error Term

To increase the precision and convergence of the proposed deep architecture, a redundant error term is used. The purpose of this redundant error term is to enhance the backpropagation of error, so that the convergence of the training error improves remarkably, as shown in Fig. 3. After adding the redundant error term, the loss function is computed by:

$$\begin{cases} \hat{L}(w,b) = L(w,b) + \tilde{L}(w,b), \\ L(w,b) = \frac{1}{2M}\sum_{m=1}^{M}\left\| h_{w,b}\left(\boldsymbol{X}^{(m)}\right) - \boldsymbol{y}^{(m)}\right\|^2, \\ \tilde{L}(w,b) = \frac{1}{2M}\sum_{m=1}^{M}\left\| h_{w,b}\left(\boldsymbol{X}^{(m)}\right) - \tilde{\boldsymbol{y}}^{(m)}\right\|^2, \end{cases} \tag{15}$$

where $\tilde{L}$ is the redundant loss term. Due to the fully connected layer before the output layer, the error term of the output layer is defined as:

$$e_i^{(8)} = -\left(y_i - a_i^{(8)}\right) \times f'\left(z_i^{(8)}\right), \quad i = \{1,2,3\}, \tag{16}$$

where $e_3^{(8)} = -\left(y_3 - a_3^{(8)}\right) \times f'\left(z_3^{(8)}\right)$ is the redundant error term that can be viewed as a feedback of the intrinsic CNN. Under this case, the error term of the fully connected layer is shown as follows:

$$e_i^{(7)} = \left(\sum_{j=1}^{S^{(8)}} W_{ji}^{(7)} e_j^{(8)}\right) f'(z_i^{(7)}). \tag{17}$$

From Eqs. (16) and (17), we find that each error term of the fully connected layer can be enhanced, that is:

$$\Delta e_i^{(7)} = W_{3i}^{(7)} e_3^{(8)} f'(z_i^{(7)}), \quad i = \left\{S^{(7)}\right\}. \tag{18}$$

Obviously, for the backpropagation algorithm, the subsequent parameters of the proposed deep architecture are increased in turn. The increasing of parameters can improve both the training error and the convergence of the training error, which will be demonstrated by the following subsection via a probabilistic representation of the proposed deep architecture.

The real-time performance of an intrusion detection mechanism has an effect on the security level of an IDS, because the response time with respect to an attack defines the security level of the IDS. The computation complexity of our method depends on the backpropagation algorithm under the proposed CNN-based deep learning architecture. Meanwhile, it is also the sum of each layer in the deep architecture. In the previous section, notation $S^{(l)}$, where $l = 1, 2, ..., 6$, denotes the set of kernels in layer l. Besides, $S^{(7)}$ and $S^{(8)}$ (it can be viewed as $l = 7$ and $l = 8$) are the sets of neurons of the relative layers. We define the number of elements in set $S^{(l)}$ as $\left|S^{(l)}\right|$. Then, the computation complexity of the fully connected layer is $O\left(\left|S^{(7)}\right|\left|S^{(8)}\right|\right)$. In the pooling layer, the computation complexity depends on the number of neurons in the following convolution layer. If we assume that the following convolution layer contains $\left|S^{(l)}\right|$ kernels and each kernel has $\left|\boldsymbol{K}^{(l)}\right|$ elements, and then the output of the pooling layer during backpropagation algorithm has $\left|S^{(l-1)}\right|$ kernels and each kernel has $\left|\boldsymbol{K}^{(l-1)}\right|$ elements (i.e., the output of the previous convolution layer). In this case, the computation complexity in a pooling layer is $O\left(\left|S^{(l)}\right|\left|\boldsymbol{K}^{(l)}\right|\right)$. Similarly, for the convolution layer, the computation complexity is $O\left(\left|S^{(l)}\right|\left|\boldsymbol{K}^{(l)}\right|\left|S^{(l-1)}\right|\left|\boldsymbol{K}^{(l-1)}\right|\right)$. Specially, for the convolution layer following the fully connected layer, we have $\left|S^{(7)}\right| = \left|S^{(6)}\right|\left|\boldsymbol{K}^{(6)}\right|$.

D. Probabilistic Representation with Redundant Error Term

Previous works have tried to depict a deep learning architecture by a complex probabilistic model [34], [35]. To demonstrate the performance of our deep architecture based on CNN for intrusion detection, we first introduce some relevant notations briefly in this paper. We assume that $\boldsymbol{X}$ and $\boldsymbol{y}$ are the training datasets and the training labels, respectively. Subsequently, $p(\boldsymbol{X})$ and $p(\boldsymbol{y})$ are the prior distributions with respect to $\boldsymbol{X}$ and $\boldsymbol{y}$, respectively. Meanwhile, the samplings of the training dataset $D = \left\{\left(\boldsymbol{X}^{(m)}, \boldsymbol{y}^{(m)}\right) \middle| \boldsymbol{X}^{(m)} \in \boldsymbol{R}^S, \boldsymbol{y}^{(m)} \in \boldsymbol{R}^L\right\}$ are independently identically distribution with the joint distribution $p_{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{y})$, where $\boldsymbol{\theta}$ denotes the parameters in the joint distribution. Then, the CNN can be defined as CNN $= \left\{\boldsymbol{x}; f_1; f_2; f_3; ...; f_I; f_{\boldsymbol{y}}\right\}$, and $f_{\boldsymbol{y}}$ is the estimator of the prior distribution $p(\boldsymbol{y})$. Besides, we denote the random variable of the hidden layer f_l by $\boldsymbol{F}_l$. According to the work in [35], [36], it is proved that the deep neural networks obey two fundamental properties, i.e., the hierarchy and generalization properties [34].

***Lemma** 1:* The whole architecture of DNNs can be formulated as a Bayesian hierarchical model. Meanwhile, each hidden layer obeys a Gibbs distribution.

The hierarchy property derives the representation of an entire CNN architecture with I hidden layers shown as:

$$\boldsymbol{F}_1 \rightarrow \boldsymbol{F}_2 \rightarrow ... \boldsymbol{F}_I \rightarrow \boldsymbol{F}_{\boldsymbol{y}}, \tag{19}$$

which is a Bayesian hierarchical model. According to Eq. (19), the CNN can be formulated as a Markov chain:

$$q\left(\boldsymbol{F}_1; \boldsymbol{F}_2; ...; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}\right) = q\left(\boldsymbol{F}_1\right) q\left(\boldsymbol{F}_1 \middle| \boldsymbol{F}_2\right) ... q\left(\boldsymbol{F}_I \middle| \boldsymbol{F}_{\boldsymbol{y}}\right). \tag{20}$$

Eq. (20) arises the destination of training the CNN, namely searching an optimal $q\left(\boldsymbol{F}_1; \boldsymbol{F}_2; ...; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}\right)$ denoted by $q^*\left(\boldsymbol{F}_1; \boldsymbol{F}_2; ...; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}\right)$ closed to the posterior distribution $p\left(\boldsymbol{F}_1; \boldsymbol{F}_2; ...; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}} | D\right)$. Generally, the distance between $p\left(\boldsymbol{F}_1; \boldsymbol{F}_2; ...; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}} | D\right)$ and $q\left(\boldsymbol{F}_1; \boldsymbol{F}_2; ...; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}\right)$ can be described by the Kullback-Leibler (KL) divergence. Therefore, the backpropagation algorithm explores the optimal distribution by minimizing the KL divergence to the posterior distribution, i.e.,

$$\begin{aligned} &q^*\left(\boldsymbol{F}_1; \boldsymbol{F}_2; ...; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}\right) = \arg\min_{q \in Q} \\ &KL\left(p\left(\boldsymbol{F}_1; \boldsymbol{F}_2; ...; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}} | D\right) \middle| q\left(\boldsymbol{F}_1; \boldsymbol{F}_2; ...; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}\right)\right). \end{aligned} \tag{21}$$

To obtain an optimal solution with lower training error, the loss function can be relaxed to:

$$\begin{aligned} &q^*\left(\boldsymbol{F}_1; \boldsymbol{F}_2; ...; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}\right) = \arg\min_{q \in Q} \\ &KL\left(p\left(\boldsymbol{F}_{\boldsymbol{y}} \middle| \{y_i\}_{i=1}^{S^{(nl)}}\right) \middle| q\left(\boldsymbol{F}_{\boldsymbol{y}} | \boldsymbol{F}_1; \boldsymbol{F}_2; ...; \boldsymbol{F}_I\right)\right), \end{aligned} \tag{22}$$

where $p\left(\boldsymbol{F}_1; \boldsymbol{F}_2; ...; \boldsymbol{F}_7; \boldsymbol{F}_{\boldsymbol{y}} | D\right) = p\left(\boldsymbol{F}_{\boldsymbol{y}} \middle| \{y_i\}_{i=1}^{S^{(nl)}}\right)$ is known for us [34].

Based on the Bayesian hierarchical model, we consider the convergence of our deep architecture. First, as is well-known,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.2990984, IEEE Transactions on Network Science and Engineering

the great generalization performance is gained by means of designing an appropriate CNN architecture and choosing a proper training dataset. Nevertheless, it has been proved that an over-parametrized CNN still acquires a desired generalization performance, and the generalization performance depends on a prior distribution of training dataset according to the Bayesian theory. In [35], the authors demonstrated the regularization property with respect to the posterior distribution of training dataset.

From Lemma 1, for the convergence of our deep architecture, the following proposition holds.

***Proposition** 1:* The convergence of the training error in the proposed deep architecture based on CNN can be enhanced by the redundant error term.

Proof: The fundamental object of a learning algorithm is learning some features of the training dataset D to model $p_{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{y})$, where $\boldsymbol{\theta}$ is the parameters of the joint distribution. Eq. (20) has given the Bayesian hierarchical model to represent a deep CNN architecture. Obviously, we find that the hidden layer f_{i-1} is the input of the subsequent hidden layer f_i. Consequently, f_{i-1} is a prior distribution of layer f_i. As we known, the Gibbs distribution is conjugate, namely if the prior and likelihood distributions are the Gibbs distribution, and then the posterior distribution also formulates the Gibbs distribution. Hence, $q\left(\boldsymbol{F}_1; \boldsymbol{F}_2; \ldots; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}\right)$ is the Gibbs distribution accordingly, which means that the Gibbs distribution holds for $p_{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{y})$. In this case, the posterior distribution $p\left(\boldsymbol{F}_1; \boldsymbol{F}_2; \ldots; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}} \mid D\right)$ can be described via an energy function, that is:

$$p\left(\boldsymbol{F}_1; \boldsymbol{F}_2; \ldots; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}} \mid D\right) = \frac{\exp\left(-E\left(\boldsymbol{X}, \theta\right)\right)}{\sum\limits_{x} \exp\left(-E\left(\boldsymbol{X}, \theta\right)\right)}, \tag{23}$$

where E is the energy function. In our deep architecture with redundant error term, the optimization model shown in Eq. (21) can be transformed into

$$\begin{aligned} &q^*\left(\boldsymbol{F}_1; \boldsymbol{F}_2; \ldots; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}^+\right) = \arg\min_{q \in Q} \\ &KL\left(p\left(\boldsymbol{F}_1; \boldsymbol{F}_2; \ldots; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}^+ \mid D, y_r\right) \middle| q\left(\boldsymbol{F}_1; \boldsymbol{F}_2; \ldots; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}^+\right)\right), \end{aligned} \tag{24}$$

where y_r and $\boldsymbol{F}_{\boldsymbol{y}}^+$ are the additional output data and the relative random variable. In the information theory, Eq. (24) can be deviated as the following form:

$$\begin{aligned} &q^*\left(\boldsymbol{F}_1; \boldsymbol{F}_2; \ldots; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}^+\right) = \arg\min_{q \in Q} \\ &\sum_{x} p\left(\boldsymbol{F}_1; \boldsymbol{F}_2; \ldots; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}^+ \mid D, y_r\right) \log \frac{p\left(\boldsymbol{F}_1; \boldsymbol{F}_2; \ldots; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}^+ \mid D, y_r\right)}{q\left(\boldsymbol{F}_1; \boldsymbol{F}_2; \ldots; \boldsymbol{F}_I; \boldsymbol{F}_{\boldsymbol{y}}^+\right)}. \end{aligned} \tag{25}$$

Multiple convolution layers can be modeled as a Gibbs distribution. Then, judging from Eq. (25), the objective function is much steeper than before, which can enhance the convergence of the optimization model.

IV. Experiments

In this section, numerical results are provided to evaluate the performance of the proposed deep architecture. The problem of a CNN-based intrusion detection mechanism is its sensitivity for various initial parameters, due to the explicit issues of a deep learning technique with respect to initialized weights and biases. Motivated by that, we first evaluate the sensitivity of our method, when different initialized weights and biases derive. Subsequently, we discuss the evaluation of our method under various strengths of attacks. Above evaluations are based

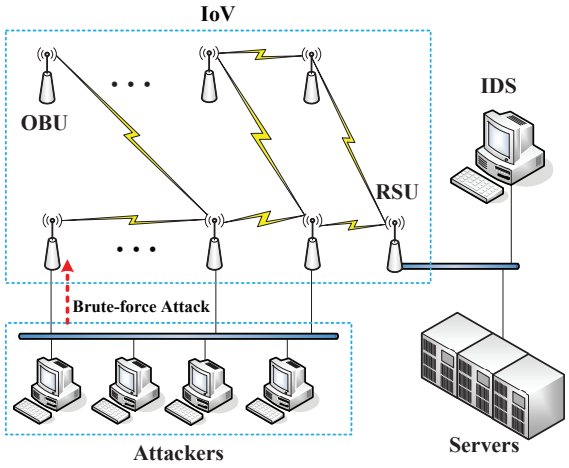

Fig. 4. Illustration of our testbed.

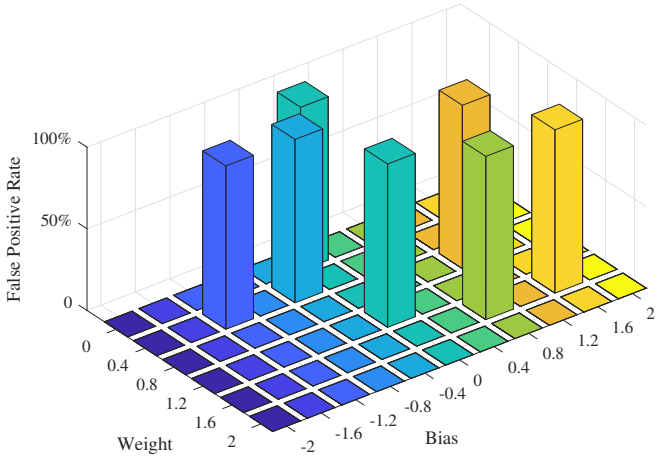

Fig. 5. Sensitivity of CNN for 12 OBUs.

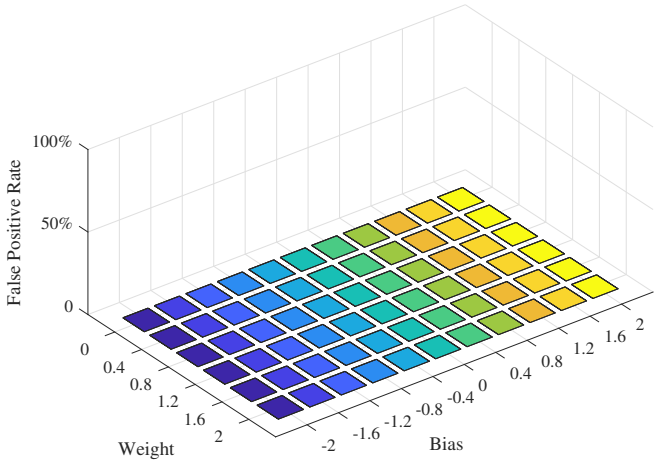

Fig. 6. Sensitivity of CNN for 30 OBUs.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.2990984, IEEE Transactions on Network Science and Engineering

on an attack. After that, we carry out our method while a series of attacks occur simultaneously with diverse persistent periods.

A. Network Traffic Dataset

We build a testbed to imitate the scene of an IoV. The testbed shown by Fig. 4 consists of 1 RSU and 30 OBUs. Furthermore, four attackers, in which the Low Orbit Ion Cannon (LOIC) is installed to implement the DDoS attack, are employed in our testbed. The LOIC implements the DDoS attack by sending a number of malicious requests with respect to the specific protocols consisting of TCP, UDP, and HTTP. In an IoV, the cost-efficient network intrusion attacks the OBU instead of the server as same as the traditional DDoS attack, since exhausting the resources of OBUs is much easier than that comparing with the server [19]. Hence, all the four attackers carry out the brute-force attack (also known as the flooding DDoS attack) to the OBUs using TCP-SYN, UDP, and HTTP floods concurrently. To imitate the traffic flows of an IoV vividly, the server provides various services for OBUs, consisting of radio, video, voice calls, and the GPS. Meanwhile, the Open Shortest Path First (OSPF) is used as the routing protocol among OBUs, and then the weights of the OSPF are defined as the general urban path loss model as shown in [37]. Under this case, we sample the packets transmitted by the link of the RSU using the Wireshark. Meanwhile, the volume of network traffic is reported over one second. Namely, each element in the dataset expresses the number of packets in the link over one second. Besides, we consider two scenarios with different network scales, in which the number of OBUs is 12 and 30, respectively. For simulations, Wollect one-week link loads from the RSU.

B. Sensitivity Analysis

This subsection first shows the sensitivity of the deep architecture for the initialized parameters (biases and weights of the convolution layer). Figs. 5 and 6 depict the false positive rates (i.e., the false alarm rate) with various initialized values. The initialized weights of convolution layers obey the symmetric uniform distribution $U(-p, p)$, and we set $p \in [0, 2]$. Fig. 5 shows the false positive rates with respect to the weights and biases for the scenario with 12 OBUs. We find that the false positive rates have explicit fluctuations for a small number of initialized parameters. There are 7 points with high false positive rates in all the 66 points with various initialized parameters. Meanwhile, from Fig. 6, it declares that the CNN architecture with redundant error term is stabilized for various initial parameters.

The LOIC provides manifold configurations for the DDoS attack, such as the strength of DDoS attack, thus we also evaluate the performance of our method under various strengths of DDoS attacks. To make the strength of DDoS attack quantifiable, we refer to a metric in this paper, which is defined as:

$$Str = \frac{X_{\text{an}} - X_{\text{n}}}{X_{max}}, \tag{26}$$

where X_{n} and X_{an} are the normal and anomalous traffic, respectively. X_{max} is the maximum traffic that can be calculated from previous known network traffic. Meanwhile,

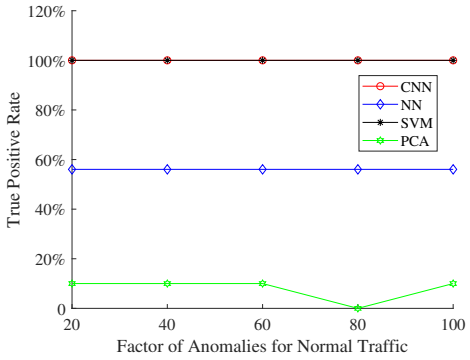

Fig. 7. True positive rates with various strengths of attacks for the scenario with 12 OBUs.

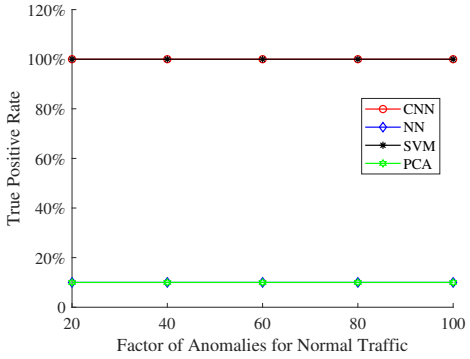

Fig. 8. True positive rates with various strengths of attacks for the scenario with 30 OBU.

three machine learning-based intrusion detection methods are brought for comparison, i.e., the traditional shallow Neural Network method (NN) [38], the Support Vector Machine method (SVM) [39], and the Principal Component Analysis method (PCA) [40]. Figs. 7 and 8 display the true positive rates with respect to various strengths of attacks for two scenarios. On the x-axis, the factors are 20, 40, 60, 80, and 100, respectively. Accoridng to the results in Figs. 7 and 8, we find that the true positive rates of four methods are consistent versus different factors except for the PCA method in the scenario with 12 OBUs. The NN, SVM and CNN methods are steady for all attacks in Fig. 7, and they are 60%, 100%, and 100%, respectively. By contrast, the PCA method has a low true positive rate when the factor is 80, and shows the lowest true positive rates among four methods. Considering the scenario with 30 OBUs as shown in Fig. 8, the true positive rate of the NN method is no longer outstanding comparing with the simulation result of the former scenario. The true positive rates of NN and PCA are 10% in this case, and they are 100% for CNN and SVM. The NN method is a shallow learning approach. When a large number of OBUs access to an RSU, the irregular fluctuations are much more obvious. As a result, these irregular fluctuations viewed as a noise for the shallow neural network arise the overfitting. Hence, the NN method shows lower true positive rates.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.2990984, IEEE Transactions on Network Science and Engineering

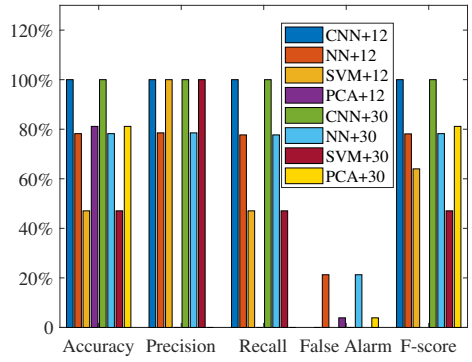

Fig. 9. Performance for 2 attacks with 2 timeslots.

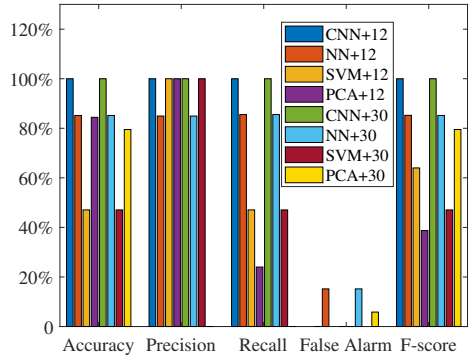

Fig. 10. Performance for 2 attacks with 3 timeslots.

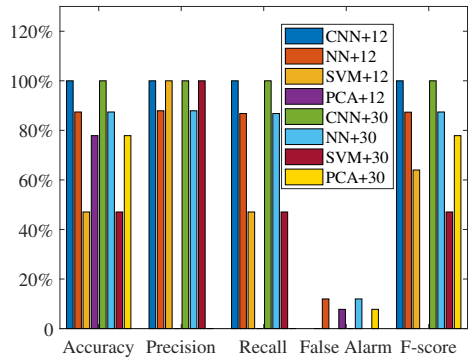

Fig. 11. Performance for 2 attacks with 4 timeslots.

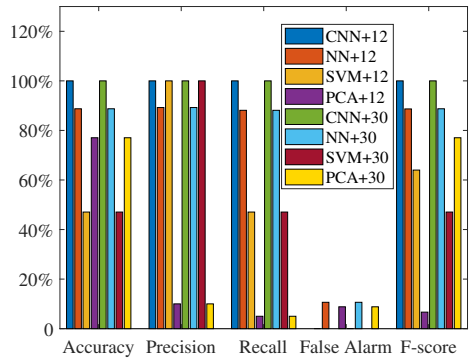

Fig. 12. Performance for 2 attacks with 5 timeslots.

C. Precision Analysis

In order to analyze the precision of our method, five metrics are used to evaluate the performance of the proposed deep architecture based on CNN. They are the accuracy, precision,

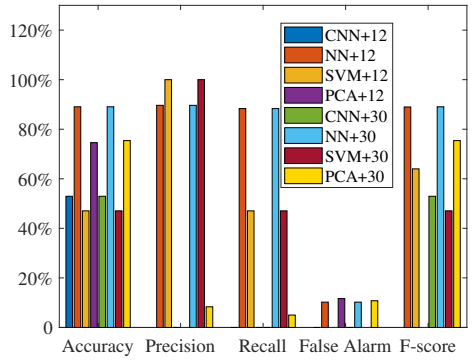

Fig. 13. Performance for 2 attacks with 6 timeslots.

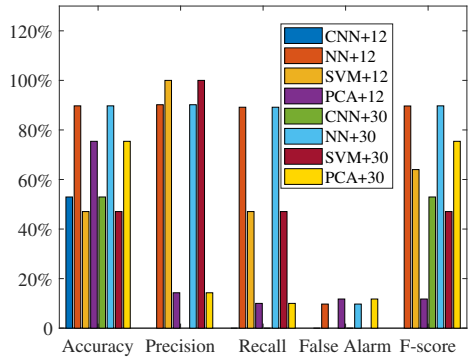

Fig. 14. Performance for 2 attacks with 7 timeslots.

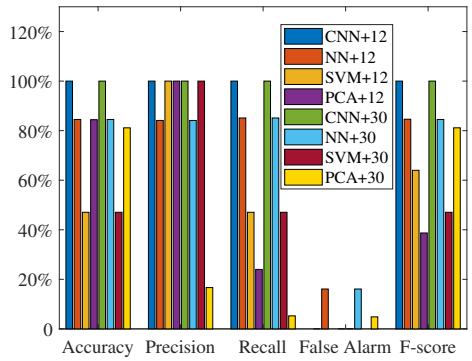

Fig. 15. Performance for 3 attacks with 2 timeslots.

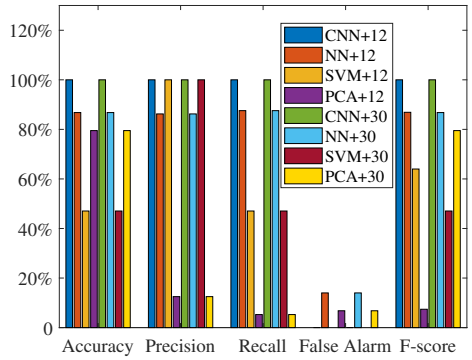

Fig. 16. Performance for 4 attacks with 2 timeslots.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2020.2990984, IEEE Transactions on Network Science and Engineering

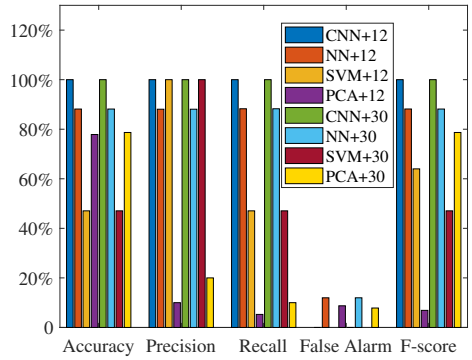

Fig. 17. Performance for 5 attacks with 2 timeslots.

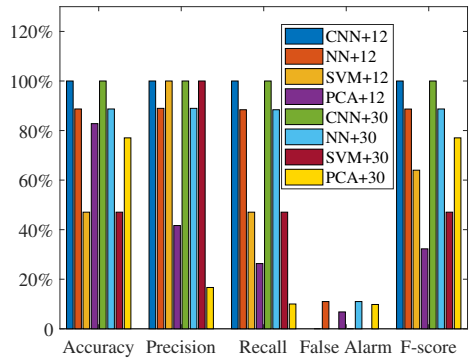

Fig. 18. Performance for 6 attacks with 2 timeslots.

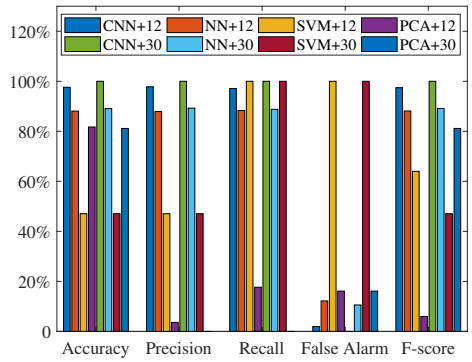

Fig. 19. Performance of CNN method for 12 nodes.

recall, false alarm and F-score, which are defined as follows:

$$\begin{cases} Accuracy = \frac{TP+TN}{TP+TN+FP+FN}, \\ Precision = \frac{TP}{TP+FP}, \\ Recall = \frac{TP}{TP+FN}, \\ False\ Alarm = \frac{FP}{FP+TN}, \\ F-score = 2 \times \frac{Precision \times Recall}{Precision+Recall}. \end{cases} \tag{27}$$

The TP, TN, FP, FN are true positive, true negative, false positive, and false negative, respectively (shown in Table I). The intrinsic definitions of the above five metrics are shown as follows:

- **Accuracy**: It illustrates the percentage of exactly detection results over the number of tests.
- **Precision**: It is defined as the percentage of the attacks detected correctly by an IDS over all of the detected attacks.
- **Recall**: It describes the percentage of the attacks detected correctly over all of the attacks.
- **False Alarm**: It shows the number of normal network traffic viewed as attack among all of normal network traffic.
- ***F*-score**: It expresses a balance between precision and recall.

TABLE I
CONFUSION MATRIX

Actual Intrusion \ Predicted Intrusion	Attack	Normal
Attack	TP	FN
Normal	FP	TN

Here we also compare our method with the NN, SVM, and PCA methods in this subsection. The NN architecture is a feedforward network with a hidden layer including 200 neurons. The input and output layers have 400 and 2 neurons, respectively. The activation function of each neuron is also the sigmoid function in the approach based on NN. Meanwhile, we used the LOIC to implement the DDoS attack using TCP-SYN, UDP, and HTTP floods. As mentioned in the above, the sensitivity analysis mainly discusses the detection performance under a single attack via 4 attackers. In this subsection, we evaluate the performance of the proposed method with respect to repeated attacks implemented by 4 attackers and various lengths of persistent period, and depict their accuracy, precision, recall, false alarm, and F-score respectively.

In Figs. 9-14, we set the number of attacks to 2, and each attack lasts for 2 to 7 timeslots. From Fig. 9, when two attacks lasting two timeslots appear, the accuracy, the precision, the recall, the false alarm, and the F-score of the NN method are 78.22%, 78.53%, 77.71%, 21.27%, and 78.12% respectively for the scenario with 12 nodes. By contrast, the proposed method is 100% in four metrics, and the false alarm is 0%. Meanwhile, they are 47.06%, 100%, 47.06%, 0%, and 64.00% for SVM. The PCA method obtains the lowest accuracy of intrusion detection. It defines two subspaces (i.e., the normal and anomalous subspaces) by means of a threshold-based approach, and then the observed traffic data are projected to two subspaces to make a decision for intrusion detection. The traffic flows in an IoV reveal so many irregular fluctuations that the normal and anomalous subspaces are blurry. As a result, the PCA method has the highest error of intrusion detection. We also obtain the same conclusion according to the scenario with 30 OBUs shown by Figs. 9- 13. Comparing with the scenario with 12 OBUs, we find that the larger scale of the network (such as 30 OBUs) has not a prominent influence on the performance.

From the other scenarios with different timeslots, we can achieve similar conclusions except for Figs. 13 and 14. When the timeslots are 6 and 7, the CNN method cannot identify the intrusion at all. The CNN method extracts the temporal and spatio-temporal features of network traffic, due to the link load of the RSU is denoted by a matrix. Hence, if the

length of an attack is large, it has a negative impact on all the convolution kernels and pooling layers. In this case, the output of the CNN-based deep architecture is chaotic. We also provide an evaluation under the scenario with multiple attacks. Figs. 15-18 display the performance with respect to the number of attacks. Obviously, the CNN and SVM methods are excellent consistently, and accuracy of the NN and PCA methods are lower comparing with the other two methods.

Finally, in Fig. 19, we execute 3000 tests to evaluate the performance of the proposed method. During each test, the number of attacks and their strength are set randomly. As shown in Fig. 19, we find that the CNN shows better performance comparing with the other three methods. The accuracy of NN, SVM, and PCA are 88.08%, 47.06%, and 81.73%, respectively, in the scenario with 12 OBUs. In contrast, it is 97.60% for the proposed deep architecture. When the number of OBUs is 30, the accuracy values of four methods are 97.60%, 88.08%, 47.06%, and 81.73%, respectively.

V. Conclusion

This paper studies the problem of intrusion detection in IoVs. The previous intrusion detection methods usually take advantage of end-to-end network traffic from OBUs to identify attacks. By contrast, we propose an intrusion detection approach based on the anomalous traffic of RSUs. In detail, we consider network resources of OBUs, and design an intrusion detection mechanism utilizing the link loads of RSUs. Moreover, a CNN-based architecture is built to extract the spatio-temporal feature of link loads. Aiming at the convergence of the deep architecture for intrusion detection, it contains a traditional CNN architecture and a redundant error term in the output layer. Subsequently, we provide a theoretical analysis of the proposed deep architecture in convergence via representing it by a Bayesian hierarchical model. Simultaneously, each layer can be modeled by a Gibbs distribution, which means that the optimal objective of the backpropagation algorithm can be converted into a posterior distribution based on an energy function. In this case, the redundant error term makes the objective function much steeper than before, thus the training error and its convergence improve dramatically. We evaluate the sensitivity and precision of our method at last, and compare it with three state-of-the-art methods.

References

[1] Z. Xiao, D. Xiao, V. Havyarimana, H. Jiang, D. Liu, D. Wang, and F. Zeng, "Toward accurate vehicle state estimation under non-gaussian noises," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 652–10 664, Dec 2019.

[2] Z. Ning, F. Xia, N. Ullah, X. Kong, and X. Hu, "Vehicular social networks: Enabling smart mobility," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 16–55, May 2017.

[3] S. Xu, Y. Qian, and R. Q. Hu, "Data-driven edge intelligence for robust network anomaly detection," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2019.

[4] D. Wang, J. Fan, Z. Xiao, H. Jiang, H. Chen, F. Zeng, and K. Li, "Stop-and-wait: Discover aggregation effect based on private car trajectory data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3623–3633, Oct 2019.

[5] J. Weng, J. Weng, Y. Zhang, W. Luo, and W. Lan, "BENBI: Scalable and dynamic access control on the northbound interface of SDN-based VANET," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 822–831, Jan 2019.

[6] P. Zhao, H. Jiang, C. Wang, H. Huang, G. Liu, and Y. Yang, "On the performance of k -anonymity against inference attacks with background information," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 808–819, Feb 2019.

[7] Z. Xiao, X. Shen, F. Zeng, V. Havyarimana, D. Wang, W. Chen, and K. Li, "Spectrum resource sharing in heterogeneous vehicular networks: A noncooperative game-theoretic approach with correlated equilibrium," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 10, pp. 9449–9458, Oct 2018.

[8] S. Prabavathy, K. Sundarakantham, and S. M. Shalinie, "Design of cognitive fog computing for intrusion detection in Internet of Things," *Journal of Communications and Networks*, vol. 20, no. 3, pp. 291–298, June 2018.

[9] J. Hong and C. Liu, "Intelligent electronic devices with collaborative intrusion detection systems," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 271–281, Jan 2019.

[10] Z. Ning, L. Liu, F. Xia, B. Jedari, I. Lee, and W. Zhang, "CAIS: A copy adjustable incentive scheme in community-based socially-aware networking," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3406–3419, 2017.

[11] H. Jiang, P. Zhao, and C. Wang, "RobLoP: Towards robust privacy preserving against location dependent attacks in continuous LBS queries," *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 1018–1032, April 2018.

[12] K. Zaidi, M. B. Milojevic, V. Rakocevic, A. Nallanathan, and M. Rajarajan, "Host-based intrusion detection for VANETs: A statistical approach to rogue node detection," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 6703–6714, Aug 2016.

[13] H. Peng and X. S. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2020.

[14] J. Zhang, X. Hu, Z. Ning, E. C. . Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2633–2645, Aug 2018.

[15] H. Sedjelmaci, S. M. Senouci, and M. A. Abu-Rgheff, "An efficient and lightweight intrusion detection mechanism for service-oriented vehicular networks," *IEEE Internet of Things Journal*, vol. 1, no. 6, pp. 570–577, Dec 2014.

[16] P. Zhao, H. Jiang, J. C. S. Lui, C. Wang, F. Zeng, F. Xiao, and Z. Li, "P3-LOC: A privacy-preserving paradigm-driven framework for indoor localization," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2856–2869, Dec 2018.

[17] X. Wang, Z. Ning, X. Hu, E. C. . Ngai, L. Wang, B. Hu, and R. Y. K. Kwok, "A city-wide real-time traffic management system: Enabling crowdsensing in social internet of vehicles," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 19–25, Sep. 2018.

[18] V. Havyarimana, Z. Xiao, A. Sibomana, D. Wu, and J. Bai, "A fusion framework based on sparse gaussian-wigner prediction for vehicle localization using GDOP of GPS satellites," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2019.

[19] N. Agrawal and S. Tapaswi, "Defense mechanisms against DDoS attacks in a cloud computing environment: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3769–3795, 2019.

[20] A. Praseed and P. S. Thilagam, "DDoS attacks at the application layer: Challenges and research perspectives for safeguarding web applications," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 661–685, Firstquarter 2019.

[21] P. Zhao, J. Li, F. Zeng, F. Xiao, C. Wang, and H. Jiang, "ILLIA: enabling k -anonymity-based privacy preserving against location injection attacks in continuous LBS queries," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1033–1042, April 2018.

[22] L. Xiao, X. Lu, D. Xu, Y. Tang, L. Wang, and W. Zhuang, "UAV relay in VANETs against smart jamming with reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4087–4097, May 2018.

[23] J. Zhou, Z. Cao, Z. Qin, X. Dong, and K. Ren, "LPPA: Lightweight privacy-preserving authentication from efficient multi-key secure outsourced computation for location-based services in vanets," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 420–434, 2020.

[24] H. Sedjelmaci, S. M. Senouci, and N. Ansari, "Intrusion detection and ejection framework against lethal attacks in UAV-aided networks: A bayesian game-theoretic methodology," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1143–1153, May 2017.

[25] O. Y. Al-Jarrah, O. Alhussein, P. D. Yoo, S. Muhaidat, K. Taha, and K. Kim, "Data randomization and cluster-based partitioning for botnet intrusion detection," *IEEE Transactions on Cybernetics*, vol. 46, no. 8, pp. 1796–1806, Aug 2016.

[26] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in Software Defined Networking," in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, Oct 2016, pp. 258–263.

[27] N. Marchang, R. Datta, and S. K. Das, "A novel approach for efficient usage of intrusion detection system in Mobile Ad Hoc Networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 1684–1695, Feb 2017.

[28] K. Huang, Q. Zhang, C. Zhou, N. Xiong, and Y. Qin, "An efficient intrusion detection approach for visual sensor networks based on traffic pattern learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 10, pp. 2704–2713, Oct 2017.

[29] Z. Sun, Y. Xu, G. Liang, and Z. Zhou, "An intrusion detection model for wireless sensor networks with an improved v-detector algorithm," *IEEE Sensors Journal*, vol. 18, no. 5, pp. 1971–1984, March 2018.

[30] Y. Hao, Q. Li, H. Mo, H. Zhang, and H. Li, "AMI-net: Convolution neural networks with affine moment invariants," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 1064–1068, July 2018.

[31] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in Convolutional Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5784–5789, Nov 2018.

[32] R. J. Cintra, S. Duffner, C. Garcia, and A. Leite, "Low-complexity approximate Convolutional Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 12, pp. 5981–5992, Dec 2018.

[33] X. Sui, Y. Zheng, B. Wei, H. Bi, J. Wu, X. Pan, Y. Yin, and S. Zhang, "Choroid segmentation from optical coherence tomography with graph-edge weights learned from deep convolutional neural networks," *Neurocomputing*, vol. 237, pp. 332 – 341, 2017.

[34] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Deep learning requires rethinking generalization," 2016.

[35] H. Steck and T. S. Jaakkola, "On the Dirichlet prior and Bayesian regularization," *NeurIPS*, 2003.

[36] X. Lan and K. E. Barner, "A probabilistic representation of deep learning," 2019.

[37] J. Andrusenko, R. L. Miller, J. A. Abrahamson, N. M. M. Emanuelli, R. S. Pattay, and R. M. Shuford, "VHF general urban path loss model for short range ground-to-ground communications," *IEEE Transactions on Antennas and Propagation*, vol. 56, no. 10, pp. 3302–3310, 2008.

[38] J. Zhao, M. Chen, and Q. Luo, "Research of intrusion detection system based on neural networks," in *2011 IEEE 3rd International Conference on Communication Software and Networks*, May 2011, pp. 174–178.

[39] S. Horng, M. Su, Y. Chen, T. Kao, R. Chen, J. Lai, and C. Perkasa, "A novel intrusion detection system based on hierarchical clustering and Support Vector Machines," *Expert Systems with Applications*, vol. 38, no. 1, pp. 306 – 313, 2011.

[40] T. Huang, H. Sethu, and N. Kandasamy, "A new approach to dimensionality reduction for anomaly detection in data traffic," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 651–665, Sep. 2016.