Research Article

Network Threat Detection Based on Group CNN for Privacy Protection

Yanping Xu ¹, **Xia Zhang**,¹ **Chengdan Lu** ², **Zhenliang Qiu**,¹ **Chunfang Bi**,³ **Yuping Lai**,⁴ **Jian Qiu**,⁵ and **Hua Zhang**⁶

¹*School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China*

²*Zhejiang Electronic Information Product Inspection and Research Institute (Key Laboratory of Information Security of Zhejiang Province), Hangzhou, China*

³*Chengzhong Primary School, Zibo, China*

⁴*School of Cyberspace, Beijing University of Post and Telecommunications, Beijing, China*

⁵*Center for Undergraduate Education, Westlake University, Hangzhou, China*

⁶*School of Computer Science, Hangzhou Dianzi University, Hangzhou, China*

Correspondence should be addressed to Chengdan Lu; lcd@zdjy.org.cn

Received 8 June 2021; Revised 15 July 2021; Accepted 3 August 2021; Published 3 September 2021

Academic Editor: Yaguang Lin

Copyright © 2021 Yanping Xu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The Internet of Things (IoT) contains a large amount of data, which attracts various types of network attacks that lead to privacy leaks. With the upgrading of network attacks and the increase in network security data, traditional machine learning methods are no longer suitable for network threat detection. At the same time, data analysis techniques and deep learning algorithms have developed rapidly and have been successfully applied to a variety of tasks for privacy protection. Convolutional neural networks (CNNs) are typical deep learning models that can learn and reconstruct features accurately and efficiently. Therefore, in this paper, we propose a group CNN models that is based on feature correlations to learn features and reconstruct security data. First, feature correlation coefficients are computed to measure the relationships among the features. Then, we sort the correlation coefficients in descending order and group the data by columns. Second, a 1D group CNN model with multiple 1D convolution kernels and 1D pooling filters is built to address the grouped data for feature learning and reconstruction. Third, the reconstructed features are input to shadow machine learning models for network threat prediction. The experimental results show that features reconstructed by the group CNN can reduce the dimensions and achieve the best performance compared to the other present dimension reduction algorithms. At the same time, the group CNN can decrease the floating point of operations (FLOP), parameters, and running time compared to the basic 1D CNN.

1. Introduction

Application scenarios for the IoT are becoming increasingly mature, which brings people to a new digital lifestyle by connecting everything [1]. However, as the IoT scope and scale continue to expand, the threat of network intrusion has become more serious than ever before [2, 3]. Malicious software, DDoS attacks, vulnerability attacks, and other attacks always occur in the IoT cyberspace, which inevitably leads to privacy leaks [4–6]. These attacks harm not only physical terminal equipment but also people's lives and property [7].

In the IoT, there are three major security and privacy challenges: terminal authentication, network attack prevention, and personal data protection [8, 9]. In terms of privacy challenges, blockchain-enabled technology using encryption algorithm will not cause privacy data leakage [10–12]. In terms of network attack prevention, network threat detection technology is required to find network intrusions and meet the demand of IoT assurance. In this situation, intrusion detection [13], malicious code detection [14], malware detection [15], malicious URL detection [16], and vulnerability mining [17] based on machine learning algorithms are

considered to be effective network threat behavior detection measures. With the upgrading of attacks and the increase in network security data, traditional machine learning methods are no longer suitable. At the same time, data analysis techniques and deep learning algorithms have developed rapidly and have been successfully applied to natural language processing, image recognition, and video detection [18, 19]. In the field of network security, many research studies have used deep learning technology to detect network threats and have garnered many achievements [13, 20].

Big data analysis techniques include oversampling imbalanced datasets, dimension reduction of high-dimensional data, and correlation analysis between features [21]. Correlation analysis studies the correlation coefficients among two or more random variables [22]. In probability theory, the correlation coefficient can reflect that there is a close relationship between variables. The range of the correlation coefficient is $[-1, 1]$. The closer the absolute value of the correlation coefficient is to 1, the closer the linear relationship between the two variables is. In contrast, the closer the absolute value of the correlation coefficient approaches to 0, the weaker the linear relationship between the two variables will be. Therefore, we use a correlation coefficient matrix to measure the relationships among the column vectors in the data matrix.

Machine learning algorithms are divided into shallow learning and deep learning [13, 22, 23]. Shallow learning is treated as a traditional machine learning technique that achieves desirable effects to address a small amount of data. Shallow learning algorithms, including support vector machines (SVMs), random forests (RFs), decision trees (DTs), and K -means algorithms, have been employed to distinguish abnormal data from network activities [13, 20]. Comparing rule-based intrusion detection systems (IDSs), shallow learning methods do not rely on the domain knowledge and can extend their generalization ability to detect the attack variants and unknown attacks. However, shallow learning is no longer suitable to address the complexity of the dataset and the diversity of the features [13]. In this situation, it emerges that deep learning is required.

Deep learning, also known as deep neural networks (DNNs), is designed from hierarchical structures composed of multiple neural layers [24]. Deep learning can extract and learn information to generate the reconstruction features from the input raw data through layer-by-layer neural processing. Benefitting from their feature reconstruction characteristics, deep learning algorithms, including CNNs, recurrent neural network (RNNs), and generative adversarial networks (GANs) [25–31] have been widely used not only for visual recognition and language understanding but also for network threat detection. Studies in [32] show that deep learning algorithm-based methods can achieve better performance when working on reconstructed features.

CNN, as one of the typical DNN models, was first proposed to solve the problem of 2D image recognition. 2D CNNs have been successfully used to learn and reconstruct features from raw data and have developed into the dominant approach for accomplishing recognition and detection tasks of image and speech analysis [33]. Due to the good characteristics of CNN learning, 1D CNN has been proposed

to address 1D signals based on 2D CNN and has achieved superior performance with high efficiency [34, 35]. To adapt to the data characteristics of 1D signals, comparing 2D CNNs, the hierarchical architecture of 1D CNNs is simplified [36]. For example, in the structure of 1D CNN, the data of the convolution kernels and pooling filters are 1D. In the structure of 2D CNN, the data of the convolution kernels and pooling filters are 2D. Therefore, in the structure and running process, 1D CNN is simpler than 2D CNN [34]. Therefore, we build a 1D CNN for analyzing network security data.

However, with the deepening of the network layers, the number of parameters increases exponentially [37]. For example, in a traditional basic 2D CNN, if the size of the input image feature is $C * H * W$, the number of convolution kernels is N , the size of each convolution kernel is $K * K$, and the size of the feature map is $M * M$. The total number of parameters in all convolution layers is $C * N * (K * K + 1) * (M * M)$. Obviously, the number of parameters is large. To reduce the number of parameters and improve the efficiency of the CNN, a group CNN is proposed to group the convolution kernels separately [38]. Suppose that the convolution kernels are divided into T groups, the number of convolution kernels in each group is N/T , the size of each convolution kernel is $K' * K'$, and the size of the feature map is $M' * M'$. The total number of parameters in the convolution layers is $C * (N/T) * (K' * K' + 1) * (M' * M')$. When grouping, the sizes of the convolution kernel and the feature maps are considered to be smaller, and the total number of parameters of all of the convolution layers is reduced. At the same time, the performance of the algorithm is improved. Therefore, we use a group CNN to address the big network data.

When analyzing the security data for network threat detection, we determined that each threat behavior had 1D characteristics, which makes the threats similar to 1D signals. Additionally, the group CNN can improve the efficiency. Therefore, learning from the successful experience of using 1D CNN to process 1D signals, we build a 1D group CNN model to perform feature learning and reconstruction of the security dataset. In this paper, we combine shadow learning and deep learning algorithms to build a network threat detection model. First, correlation coefficients are computed to measure the relationships of the features. Then, we sort the correlation coefficients in descending order and group the data by the columns. Second, a 1D group CNN model with multiple 1D convolution kernels and 1D pooling filters is built for feature learning and reconstruction. In each convolution layer and pooling layer, the convolution kernels and pooling filters are grouped. Third, the reconstructed features are input to the shadow learning models for threat prediction.

The proposed method includes the following advantages:

- (1) Compared with the traditional basic 1D CNN, the proposed group CNN model with grouped convolution kernels and pooling filters reconstructs the features layer by layer and reduces the FLOP, parameters, and running time
- (2) The proposed data grouping, which is based on correlation coefficients between the features, can

enhance the structural information used by group CNN to address the data

- (3) The proposed group CNN model can reduce the dimensions by generating fewer reconstructed features and can achieve high performance
- (4) The FLOP and parameter counts of the group CNN are calculated and are less than those of the basic 1D CNN

The remainder of this paper is organized as follows. Section 2 discusses the related work using shallow learning algorithms and deep learning technology in network threat detection. A description of the 1D group CNN model is provided in Section 3. Experimental results and analysis are presented in Section 4. The work is concluded in Section 5.

2. Related Work

Machine learning techniques, including shallow learning and deep learning algorithms, have been used for anomaly detection since the early 2000s and can automatically mine hidden information on the differences between normal and malicious behaviors.

Shallow learning algorithms, such as traditional machine learning algorithms, were previously applied to analyze system logs, malicious software, and network traffic and to output the predicted labels of the input data. By comparing the predicted labels with the true labels, the performance of the shallow learning algorithms can be achieved. The most widely used algorithms include SVM, DT, NB, and K-means [39, 40]. Buczak et al. [39] provided a summary as a survey to describe some machine learning and data mining methods, such as DT, SVM, RF, and NB, which were used for cybersecurity intrusion detection. Kruczkowski and Szykiewicz [41] used SVM with kernels to build a malware detection model. The results revealed that SVM was a robust and efficient method for data analysis and it increased the efficiency of malware detection. Bilge et al. [42] presented the EXPOSURE system to analyze large-scale and passive domain name service (DNS) data. The classifier is built by J48 DT. The experimental results suggested that the minimum error was achieved by a decision tree. Aung and Min [43] used K-means and classification and regression tree (CART) algorithms to mine the KDD'99 dataset for intrusion detection. The experimental results showed that the hybrid data mining method could achieve good accuracy in performance analysis with time complexity. Mo et al. [44] discussed three data clustering algorithms, including K-means, fuzzy C means (FCM), and expectation maximization (EM), to capture abnormal behavior in communication networks. The experimental results showed that FCM was more accurate.

More recently, deep learning technology is developing rapidly and has been successfully been applied to a variety of tasks, such as natural language processing, image recognition, and computer vision [45]. CNNs, as typical DNN models, have feedforward neural networks with convolution calculations and deep structures, which can learn and reconstruct features more accurately and efficiently. According to

the type of raw data, 1D CNN and 2D CNN models should be built. A 1D CNN is constructed to process one-dimensional sequence signal data and natural language, and a 2D CNN is constructed to address two-dimensional image and video data [36]. Because the CNN can learn and reconstruct features, both 1D CNN and 2D CNN are used for network threat detection.

Xiao et al. [46] proposed a network intrusion detection model based on a CNN. The original traffic data were reduced in dimensions through principal component analysis (PCA) and an autoencoder (AE), and then, the data were converted into a 2D image format. Next, the 2D data were input to the CNN model to evaluate the performance. Wang et al. [47] proposed a method that represented raw flow data as an image and used the CNN for classification and identification without manually selecting and extracting features. Experimental results showed that this method had high availability and high accuracy in malicious traffic identification. Zhang et al. [48] proposed a feature-hybrid malware variant detection approach based on 2D CNN and 1D BPNN. A 2D CNN was designed to compute the dot product and compress the dimension of the PCA-initialized opcode matrix. Experimental results showed that the method achieved more than 95% malware detection accuracy. Zhang et al. [49] proposed converting opcodes into a 2D matrix and adopted the CNN to train the 2D opcode matrix for malware recognition. Experimental results showed that their approach could significantly improve the detection accuracy by 15%. Yan et al. [50] proposed converting Android opcode into 2D gray images with fixed size and adopted a CNN to train and detect Android malware. Through the above literature, we can determine that the input data of the 2D CNN model must be converted into 2D data first.

Ma et al. [51] proposed a hybrid neural network comprised of 1D CNN and DNN to learn the characteristics of high-dimension network flows for network anomaly detection. Experimental results showed that the proposed method was better than those of other algorithms on the comprehensive performances. Azizjon et al. [52] proposed a 1D CNN model to serialize the TCP/IP packets in a predetermined time range as an invasion Internet traffic model for the IDS. Experimental results showed that 1D CNN and its variant architectures had the capability to extract high-level feature representations and outperformed the traditional machine learning classifiers. Wei et al. [53] proposed a 1D CNN-based model to identify phishing websites on a URL address text, which was converted to one-hot character-level representation. This mode liked the 1D CNN to analyze natural language. Experimental results showed that the method was faster to detect zero-day attacks. Zhang et al. [54] designed a flow-based intrusion detection model called SGM-CNN, which first integrated SMOTE and GMM to make an imbalanced class process and used 1D CNN to detect the network traffic data with high accuracy. Experimental results showed that SGM-CNN was superior to the state-of-the-art methods, and effective for large-scale imbalanced intrusion detection.

The group convolution was first proposed and used in AlexNet by Krizhevsky et al. [37] for distributing the model over two GPUs to handle the memory insufficient issue.

FIGURE 1: The architecture of the 1D group CNN.

AlexNet was designed as a the group convolution method could increase the diagonal correlations between the convolution kernels, reduce the training parameters, and be not easy to overfit. Zhang et al. [38] proposed interleaved group CNNs called IGCNets, which contained a pair of successive interleaved group convolutions, i.e., the primary group convolution and the secondary group convolution. IGCNets was wider than a regular convolution. Experimental results demonstrated that IGCNets was more efficient in parameters and computation complexity. Xie and Girshick [55] proposed a simple, highly modularized network architecture named ResNeXt, which was based on AlexNet and constructed by repeating a building block. The idea of ResNeXt was consistent of group convolutions. Without increasing the complexity of the parameters, the accuracy of the model could be improved, and the number of super parameters could be reduced. Lu et al. [39] proposed a novel repeated group convolutional kernel (RGC) to remove the filter's redundancy from group extent. SRGC-Nets worked well in not only reducing the model size and computational complexity, but also decreasing the testing and training running time.

In the 2D CNN-based model, the input data are converted to the image format. In the 1D CNN-based model, the input data are treated as timing serial signals, similar to natural language. Compared with a 2D CNN, the structure of a 1D CNN is simpler, which makes the computational complexity lower. Therefore, we intend to learn from the experience of applying the 1D CNN to address the data and to construct a network threat detection model for feature learning and reconstruction.

3. Proposed Solution

The architecture of the proposed network threat detection model, which combines the 1D group CNN algorithm and machine learning classification methods, is shown in Figure 1. First, correlation coefficients are computed to measure the relationships between the features. Then, we sort the correlation coefficients in descending order and group the data. Second, a group CNN model with multiple groups of

convolution kernels and pooling filters is built for feature learning and reconstruction. In the group CNN model, the input data are divided into multiple groups. Similarly, convolution kernels and pooling filters in each layer are divided into multiple groups. Each group of data is computed by each convolution kernel and is then computed by each pooling filter.

Finally, a concatenating layer is used to concatenate multiple groups of data to form one group of reconstructed data. Third, the reconstructed data are input to the shadow machine learning model for threat prediction. In the shadow machine learning model, traditional machine learning algorithms are used to identify normal or abnormal samples from reconstruction data. Then, the accuracy, precision, recall, and F1, which are the detection performance indicators, are computed according to the statistics of the confusion matrix.

3.1. Group Convolutional Neural Network for Feature Reconstruction. The convolutional neural network (CNN) is one of the representative algorithms for deep learning. It is a type of deep feed forward neural network that has convolution calculations [56]. CNNs have the capability of representation learning to generate reconstruction features. At the same time, by the convolution operation and pooling operation, a CNN can achieve the purpose of reducing the dimensions of the input data [57]. Additionally, grouped convolution kernels and pooling filters can reduce the number of parameters and improve the performance [39]. Therefore, we use 1D group convolution kernels to build a 1D group CNN model in this work.

The 1D group CNN includes multiple convolutional layers, multiple pooling layers, a full connection layer, a concatenating layer, and an output layer. In each convolutional layer, the convolutional kernels are divided into multiple groups. At the same time, in each pooling layer, the pooling filters are divided into multiple groups. The fully connected layer determines the dimensions of the reconstruction features of each group. The concatenating layer is used to concatenate the reconstruction features of each group to form the final results. The combination of multiple layers makes the group CNN output the low-dimensional

reconstruction features, which can not only strengthen the original data's features but also relatively reduce the dimension.

3.2. Feature Correlation. In this work, we assume that the input data are $X = (x_1, x_2, \dots, x_n, \dots, x_N)$, $x_n = (x_{n1}, x_{n2}, \dots, x_{nD})^T$, $n = (1, 2, \dots, N)$, containing N independent D -dimensional samples. Usually, the malicious samples have some similar values of the same features and so are the benign samples. Thus, there are certain correlations between the features and the labels. We calculate the correlations between the data features and labels based on the correlation coefficients.

First, we calculate the correlations between the data features and labels based on the correlation coefficients to form a correlation coefficient matrix $\mathbf{R}$. Then, we randomly select one row vector $\mathbf{R}_i$ and rank the correlation elements in descending order. Furthermore, we divide the data into T groups by columns equally according to the descending correlations. Usually, each group has the same number of features, which is D/T . The input data in the t th ($0 < t \leq T$) group are expressed as $X_t = (x_{1,t}, x_{2,t}, \dots, x_{n,t}, \dots, x_{N,t})$. So the correlation coefficients of the first group data are the biggest, and that of the last group data are the smallest.

3.3. Group CNN. After the data are grouped, we start to establish the group CNN model, which contains L convolution layers, L pooling layers, a full connection layer, a concatenating layer, and an output layer. Like the group counts of the input data, the convolution kernels and pooling filters in each layer are also divided into T groups. Further, there are M convolution kernels in each group.

Suppose that the m th ($0 < m \leq M$) convolution kernel in the t th ($0 < t \leq T$) group of the l th ($0 < l \leq L$) convolution layer is expressed as $K_l^{m,t}$. Convolution operations are conducted between the grouped data X_t and the convolution kernel, or the output $R_{l-1}^{m,t}$ of the previous pooling layer. Then, activation function is working to generate the feature maps. Suppose the feature map in the t th group of the l th convolution layer by the m th convolution kernel is $S_l^{m,t}$, which is expressed as follows:

$$S_l^{m,t} = \begin{cases} \text{Re } LU(\text{conv1D}(K_l^{m,t}, X_t) + b_l^{m,t}), & l = 1, \\ \text{Re } LU(\text{conv1D}(K_l^{m,t}, R_{l-1}^{m,t}) + b_l^{m,t}), & 1 < l \leq L, \end{cases} \quad (1)$$

where $\text{Re } LU(\cdot)$ is the nonlinear activation function. $\text{conv1D}(\cdot)$ is the 1D convolution function. $R_{l-1}^{m,t}$ is the output of the m th pooling filter in t th group of the $(l-1)$ th pooling layer. $b_l^{m,t}$ is the bias of the t th group in the l th convolution layer.

After the convolution layer, a pooling layer not only reduces the dimensions of feature maps from the upper convolution layer to reduce the computational cost but also provides basic translation invariance. The l th pooling layer is immediately after the l th convolution layer. Suppose the m th pooling filter of the t th group in the l th pooling layer is $P_l^{m,t}$. The input data of the l th pooling layer is the output of the l th convolution layer, and the output data of the t th group

in the l th pooling layer is $R_l^{m,t}$, which is expressed as follows:

$$R_l^{m,t} = \text{Re } LU(\max \text{ pooling}(S_l^{m,t}, P_l^{m,t})), \quad (2)$$

where $\max \text{ pooling}(\cdot)$ is the pooling function. The max pooling is adopted in this paper.

After the last pooling layer is the full connection layer. Last pooling layer is connected to a fully connected layer. After the convolution operations and pooling operations, the original data is converted into the feature maps. In the full connection layer, the t th feature map is mapped to the group reconstruction features X_t' by a global convolution operation:

$$X_t' = \text{Re } LU\left(\sum_m (\text{conv1D}(K_{full}^{m,t}, R_L^{m,t}) + b_L^{m,t})\right), \quad (3)$$

where $K_{full}^{m,t}$ is the convolution kernel of the full connection layer. $b_L^{m,t}$ is the bias of the full connection layer.

Further, the fully connected layer is connected to the concatenating layer. The T groups of the reconstructed features X_t' are concatenated to form the final reconstructed features X' :

$$X' = \text{concatenate}(X_t'), \quad (4)$$

where $\text{concatenate}(\cdot)$ is the reconstruction features' concatenated function.

The size of X' is $N \times D'$. When D' is less than D , it means that the dimension of D' is less than that of D . In other words, 1D CNN realizes the generation of reconstruction features and the dimension reduction of features.

3.4. Floating Point of Operations and Parameters. Floating point of operations (FLOP) is used to calculate the times of multiplications and additions, which are related to the overall running time of the model [58]. In this section, we want to calculate the FLOP and parameter counts of the group CNN. However, the group CNN is proposed on the basic 1D CNN. So, we first calculate the FLOP and parameter counts of the basic 1D CNN. Then, we calculate the FLOP and parameter counts of the group CNN based on that of the basic 1D CNN.

3.4.1. FLOP and Parameter Counts of the Basic 1D CNN. Suppose that the basic 1D CNN with fully connected layers is used for feature reconstructed. First, FLOP is computed. We assume that the input data are X , containing N independent D -dimensional samples. In the basic 1D CNN, the number of the input convolution channels is C_{in} , the number of the convolution kernels is M' , and the size of the convolution kernels is $1 * W'_1$. The size of the feature map of the convolution operation is $1 * W'_2$. The numbers of the output convolution channels are C_{out} . The FLOP performed by a

convolution layer is as follows:

$$N * C_{in} * M' * (1 * W'_1 + 1) * W'_2 * M' * W'_1 * C_{out}, \quad (5)$$

where $(1 * W'_1 + 1)$ means that a multiplication is performed by one convolutional kernel sampling the input data. $(+1)$ is to add the bias.

$*W'_2$ means the number of multiplications performed by one convolutional kernel to get the feature maps of the output convolution operation. The definition of W'_2 is $W'_2 = (D + 2padding - W'_1)/stride + 1$, where $padding = 0$, $stride = 1$.

$*M'$ means multiple convolutional kernels computing in the operation.

$*M' * W'_1$ means the number of addition from the feature map of the convolution operation to the output feature map of the convolution layer.

It is noted that the operations of $Re\ LU(\cdot)$ and the pooling layers do not contain multiplication and addition operations. Therefore, the FLOP does not consider the operations of $Re\ LU(\cdot)$ and the pooling layers.

$*C_{in}$ and $*C_{out}$ means repeating calculation in multiple input channels and output channels.

$*N$ means repeating calculation of all the samples.

Basic 1D CNN has L convolution layers, so the FLOP of the basic 1D CNN model equals the sum of the FLOP of each convolution layer, which can be computed as follows:

$$\sum_{l=1}^L N * C_{l,in} * M'_l * (1 * W'_{l,1} + 1) * W'_{l,2} * M'_l * W'_{l,1} * C_{l,out}. \quad (6)$$

Then, the bias term is ignored and the FLOP calculation formula (6) is written as follows:

$$o\left(\sum_{l=1}^L N * C_{l,in} * M'^2_l * W'^2_{l,1} * W'_{l,2} * C_{l,out}\right). \quad (7)$$

It can be seen that FLOP is determined by the number of the samples, the number of the convolutional layers, the number of the convolutional kernels per layer, the size of each convolutional kernel, the length of the feature map of the convolution operation, and the number of the input and output convolution channels.

Next, we computed the parameter count of basic 1D CNN. The parameter count is to get the statistics of the parameters during the basic 1D CNN operating, containing weighting parameters and bias parameters, which appear in the running process of the model. In the above basic 1D CNN, in the case of a single channel and a single convolution kernel, the number of the parameters is $(W'_1 + 1)$. When the number of the convolution kernels is M' and the number of the convolution layers is L , the parameter count of each layer is $\sum_{l=1}^L N * C_{l,in} * M'_l * (W'_{l,1} + 1) * C_{l,out}$. Then, the bias term is ignored and the parameter count calculation formula is

written as follows:

$$o\left(\sum_{l=1}^L N * C_{l,in} * M'_l * W'_{l,1} * C_{l,out}\right). \quad (8)$$

It can be seen that the parameter count is determined by the number of the samples, the number of the convolutional layers, the number of the convolutional kernels per layer, the size of each convolutional kernel, and the number of the input and output convolution channels.

3.4.2. FLOP and Parameter Counts of the Group CNN. Like basic 1D CNN, the FLOP and parameter count of group CNN can be computed. Suppose that the input data is X , containing N independent D -dimensional samples, which are grouped to T groups. It means that the dimension of each group data is D/T . The numbers of the input and output convolution channels are C_{in} and C_{out} . The structure of group CNN contains L convolution layers and L pooling layers. There are T group convolution kernels in each convolution layer. The pooling layer is the same. There are M convolution kernels in each group convolution kernels. The size of each convolution kernel is $1 * W_1$. The size of the feature map of the convolution operation is $1 * W_2$. Therefore, the FLOP of each group is

$$N * C_{in} * M * (1 * W_1 + 1) * W_2 * M * W_1 * C_{out}, \quad (9)$$

where $W_2 = ((D/T) + 2padding - W_1)/stride + 1$, where $padding = 0$, $stride = 1$.

Total FLOP of the model equals the sum of the FLOP of each convolution layer, which can be computed as follows:

$$\sum_{l=1}^L \sum_{t=1}^T N * C_{l,t,in} * M_{l,t} * (1 * W_{l,t,1} + 1) * W_{l,t,2} * M_{l,t} * W_{l,t,1} * C_{l,t,out}. \quad (10)$$

Then, the bias term is ignored and the FLOP in formula (6) is optimized as follows:

$$o\left(\sum_{l=1}^L \sum_{t=1}^T N * C_{l,t,in} * M^2_{l,t} * W^2_{l,t,1} * W_{l,t,2} * C_{l,t,out}\right). \quad (11)$$

Similarly, the parameter count of group CNN can be computed as follows:

$$o\left(\sum_{l=1}^L \sum_{t=1}^T N * C_{l,t,in} * M_{l,t} * W_{l,t,1} * C_{l,t,out}\right). \quad (12)$$

It can be seen that the FLOP and parameter count are determined not only by the number of the samples, the number of the convolutional layers, the number of the convolutional kernels per layer, the size of each convolutional kernel, and the number of the input and output convolution channels, but also by the number of groups.

Now, let us compare the FLOP and parameter count of group CNN with that of basic 1D CNN. From formula (7), formula (8), formula (11), and formula (12), we can find that there are many parameters to decide the FLOP and parameter count. We cannot compare them directly. But we can assume some comparison conditions. Because the length of input data in group CNN to that of basic 1D CNN is $1/T$, we assume that the length of convolutional kernels in each layer of group CNN to that of basic 1D CNN is $1/T$, that is, $W'_1 = T * W_1$. Similarly, $W'_2 \approx T * W_2$. According to the comparison of formula (7) and formula (11), it can roughly be seen that the FLOP of group CNN is smaller than that of 1D CNN. Similarly, according to the comparison of formula (8) and formula (12), it can roughly be seen that the parameter count of group CNN is smaller than that of 1D CNN. Actually, in experiments, we set completely different values of the parameters for the two models to achieve the best feature representation effect. More specifically, a comparison of the results are seen in Section 4.3.5.

3.5. Shallow Machine Learning Classifier. Shallow machine learning has good performance and high efficiency. Therefore, in this work, we use SVM as a shallow machine learning algorithm to build the classification model and identify the malicious samples in the dataset.

Shallow machine learning is consisted of two stages: training stage and testing stage [59]. In the training stage, the high-dimensional original dataset is reconstructed to the low-dimensional features by the training of the group CNN. Then, the dataset containing low-dimensional reconstruction features is input to the shallow machine learning classifier to train and obtain the optimal model structure. In the testing stage, the high-dimensional original testing dataset is input to the trained group CNN model to obtain the low-dimensional reconstructed features [60, 61]. Then, the dataset containing low-dimensional reconstructed features is input to the trained shallow machine learning classifier to get the labels of the predicted testing data.

In the experiment, the true labels of the testing dataset have been known, so the performance of the shallow machine learning models, such as accuracy, precision, recall, and F1, can be obtained by comparing the true labels with the predicted labels and calculating the confusion matrix.

The confusion matrix for binary classification includes four index items, such as true positive (TP), false negative (FN), false positive (FP), and true negative (TN). Then, other evaluation metrics as performance are defined as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}, \quad (13)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad (14)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad (15)$$

TABLE 1: Details of the datasets.

Dataset	Number of features	Number of samples (normal/abnormal)
KDDCUP99	41	10200 (5000/5200)
CICMalDroid2020-139	139	3795 (1795/2100)
CICMalDroid2020-470	470	3795 (1795/2100)

$$F1 = \frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FN} + \text{FP}} (\beta = 1). \quad (16)$$

4. Experiments

4.1. Dataset. The data come from the public datasets in cyberspace and contain the data of the network threat behavior. The details of the datasets are shown in Table 1.

KDDCUP99 [61] is the most famous and frequently cited dataset on intrusion detection. The whole dataset is very big and classified to 5 classes. In our work, we just randomly extract a small part, and only use them in 2 classes consisting of the normal and abnormal samples. The data set contains 41 features, which are divided into 4 categories: 9 basic features of the TCP, 13 content features of the TCP, 9 statistical features of the traffic based on time, and 10 statistical features of the traffic based on host.

CICMalDroid2020 [62] is downloaded from the website of Canadian Institute for Cybersecurity datasets. The original dataset contains 5 categories of Android samples. In our work, we just use the whole banking dataset, which contains 2100 malware samples, and the whole benign dataset, which contains 1795 benign samples. CICMalDroid2020-139 consists of 139 extracted features including the frequencies of system calls. CICMalDroid2020-470 consists of 470 extracted features including the frequencies of system calls, binders, and composite behaviors.

For most machine learning-based classification tasks, imbalanced datasets could cause the classification surfaces of the classifiers bias to the majority class, which leads to the misclassification of the minority class. Generally, the network threat data is treated as the minority class. Therefore, in our experiment, the ratios of “Normal” and “Abnormal” instances in all the three datasets are close to 1, which can void the imbalanced problem.

4.2. Machine Learning Classifiers. There are many shallow machine learning classifiers, e.g., NB, RF, and LR. Through our previous experimental results and analysis of the existing literature, we find that SVM is the most commonly used classifier.

SVM has many advantages: (1) It has good stability, which in many cases can maintain good classification performance. (2) It can deal with the noise and outlier data well by introducing relaxation variable. (3) It can effectively solve the problem of nonlinear and high-dimensional data. (4) It can keep good classification efficiency and effect for small data sets.

FIGURE 2: The performance of the reconstruct features at different ratios.

To sum up, combined with the characteristics of our dataset, which is high-dimensional and small, we choose SVM as the classifier in our experiment.

All experiments are performed in JetBrains PyCharm 2017 with Python 3.6 interpreter on a laptop Intel CORE i5-6200U 2.3 GHz with 8 GB RAM running the Windows 10 OS.

4.3. Experiment Results and Discussion

4.3.1. Comparison of the Reconstructed Features. In this section, the performances of the reconstructed features at different ratios are compared. According to the output size of the fully connected layer, the dimensions of the reconstructed features are different. In this section, to identify the performance of the reconstructed features, the lengths of the reconstructed features are set according to different situations. Specifically, the ratios of the reconstructed feature length to the original data length are set to 5%, 10%, 15%, 20%, 25%,

and 30%. First, the original data are input to group CNN models to generate the reconstructed features. Second, the data composed of reconstructed features are input to SVM, and then, the accuracy, precision, recall, and F1 are computed to evaluate the performance of the reconstructed features. The performances of the reconstructed features at different ratios are plotted in Figure 2. In addition, it should be noted that the number of iterations of the group CNN algorithms is 1000. The recorded results are the average of 5 experiments.

According to the curve of the performance of the reconstructed features at different ratios in Figure 2, including the accuracy, precision, recall, and F1, we can obtain some conclusions. First, the performances of the reconstructed feature data at some low ratios are better than those of the original data, whose ratio is 100%. In particular, the performances of the KDDCUP99 dataset are more obvious. Therefore, it is necessary to reduce the data dimensions by using the group CNN to reconstruct the features, which cannot reduce the

TABLE 2: The parameters of group CNN and basic 1D CNN network structures.

	KDD99	Basic 1D CNN		CICMalDroid2020-470	KDD99	CICMalDroid2020-139	Group CNN	
		CICMalDroid2020-139	CICMalDroid2020-470				CICMalDroid2020-139	CICMalDroid2020-470
Count of the convolution layers	3	4	5	3	4	5		
Count of the pooling layers	0	3	2	0	3	2		
Learning rate	0.0009	0.0009	0.0009	0.0009	0.0009	0.0009		
Count of groups	1	1	1	2	2	4		
Stride	1	1	1	1	1	1		

TABLE 3: Accuracy of group CNN and basic 1D CNN.

(a) Accuracy of basic 1D CNN

Datasets	5%	10%	15%	Accuracy 20%	25%	30%	100%
KDD99	0.9605	0.9566	0.9599	0.9627	0.9667	0.9548	0.8788
CICMalDroid2020-139	0.7760	0.7854	0.7721	0.7587	0.7733	0.7724	0.7094
CICMalDroid2020-470	0.6783	0.7065	0.6808	0.6320	0.6127	0.6246	0.7171

(b) Accuracy of group CNN

Datasets	5%	10%	15%	Accuracy 20%	25%	30%	100%
KDD99	0.9612	0.9717	0.9724	0.9764	0.9696	0.9692	0.8788
CICMalDroid2020-139	0.7681	0.7988	0.8091	0.7945	0.7967	0.7891	0.7094
CICMalDroid2020-470	0.8111	0.8058	0.7937	0.8108	0.7985	0.7983	0.7171

data quality. Second, the higher the dimension of the original data is, the lower the ratio of the reconstruction features with better performance.

For example, KDDCUP99 is a low-dimension dataset, whose highest accuracy and F1 are at 15%. CICMalDroid2020-139 is a middle-high-dimensional dataset, whose highest accuracy and F1 are at 10%. Meanwhile, CICMalDroid2020-470 is a high-dimensional dataset, whose highest accuracy and F1 are at 5%. To sum up, we can conclude that reconstructed features are helpful to reduce the data dimensions and improve the performance.

4.3.2. Comparison of the Group CNN and the Basic 1D CNN.

Both the group CNN and the basic 1D CNN can reconstruct features. In this part, we compare the performance of the reconstructed features by these two methods. First, the original data are input to group CNN and basic 1D CNN models, respectively. Different ratios from 5% to 30% of the reconstructed features are generated. Second, the data composed of reconstructed features are input to SVM, and the accuracy are computed to evaluate the performance of the reconstructed features. The parameters of their network structures are shown in Table 2. The performance of different ratios of the reconstructed features are recorded in Table 3. In addition, it should be noted that the number of iterations of the CNN algorithms is 1000. The recorded results are the average of 5 experiments.

The original data are directly input to SVM, and the accuracy is recorded in the last column of Tables 3(a) and 3(b). By contrast, the accuracy at different ratios from 5% to 30% of the reconstructed features are recorded in other columns. Comparing the results in Table 3(a), we find that in some situations the accuracy of the reconstructed features by the basic 1D CNN is higher than that of the original data. KDDCUP99 achieves the highest accuracy at 25%. CICMalDroid2020-139 achieves the highest accuracy at 10%. And CICMalDroid2020-470 achieves the highest accuracy with the original data. Comparing the results in Table 3(b), we find the accuracy of the reconstructed features

by the group CNN is higher than that of the original data. KDDCUP99 achieves the highest accuracy 0.9764 at 15%. CICMalDroid2020-139 achieves the highest accuracy 0.8091 at 10%. And CICMalDroid2020-470 achieves the highest accuracy 0.8111 at 5%. Comparing the results in Table 3(a) with that in Table 3(b), we find that the accuracy by the group CNN is generally higher than that by the basic 1D CNN. And the highest accuracy of each dataset in Table 3(b) by group CNN is higher than that in Table 3(a) by the basic 1D CNN. Furthermore, the datasets get the highest accuracy by the group CNN at the lower ratios. For example, KDDCUP99 gets the highest accuracy by the group CNN at 15%, but gets the highest accuracy by basic 1D CNN at 25%. Finally, we can conclude that the performance of the group CNN is better than that of basic 1D CNN mainly because grouped data based on the feature correlation helps to improve the inside stickiness of the data of each group.

4.3.3. Training Loss of the Group CNN. During training stage, the training loss is achieved based on the cross entropy loss function to compare the probability that the predicted labels of the reconstructed features are close to the real labels. The smaller the training loss is, the closer the predicted labels to the true labels of each data. In this section, we study the trend of the training loss of the group CNN. KDDCUP99 and CICMalDroid2020-139 are grouped to two groups, while CICMalDroid2020-470 is grouped to four groups. The grouped data are separately input to the group CNN to train the models. Then, different ratios from 5% to 30% of the reconstructed features are generated. During the training of the group CNN, the loss of each iteration is recorded and plotted in Figure 3. The number of iterations in the training stage is 1000.

From the curves in Figure 3, on the one hand, we find that some training loss curves of the grouped data are closer to each other and approaching to 0. For example, in Figure 3(a), the training loss curves of 20% reconstructed feature data of KDDCUP99, which are grouped to two groups, are closer. So are the training loss curves of 15%

(a) Training loss of KDDCUP99 by group CNN

FIGURE 3: Continued.

(b) Training loss of CICMalDroid2020-139 by group CNN

FIGURE 3: Continued.

FIGURE 3: Training loss of group CNN.

reconstructed feature data of CICMalDroid2020-139 in Figure 3(b), and the training loss curves of 5% reconstructed feature data of CICMalDroid2020-470 in Figure 3(c).

Furthermore, the ratios of the closer training loss curves in Figure 3 are the same as that of the highest accuracy in Table 3(a). On the other hand, we find that when the curves converge, the training loss curve of group 1 is under that of group 2 in Figures 3(a) and 3(b), and the loss curves are the same in Figure 3(c), where the loss curve of group 1 is at the bottom and the loss curve of group 4 is on the top. That is because the data are grouped based on the feature correlation. We first calculate the correlations between features, and rank the correlations in descending order. Then, we divide the data into several groups equally according to the descending correlation coefficients. So, the correlation coefficients of the first group are biggest, and that of the last group are smallest. Therefore, the loss of reconstructed features are smaller when the correlation coefficients are larger.

4.3.4. Comparison of the Dimension Reduction Algorithms. The group CNN can reconstruct features and reduce the dimensions of the features. Therefore, the group CNN can be seen as a dimension reduction algorithm. At present, there are many dimension reduction algorithms, such as PCA, FA, ICA, and SVD. In this section, we choose PCA and SVD to

compare with the basic 1D CNN and the group CNN. Like in Section 4.3.1, first, the dimensions of the original data by the dimension reduction algorithms are reduced to 5%, 10%, 15%, 20%, 25%, and 30%, separately. Then, the dimension reduction data are input to SVM. Accuracy and F1 are calculated to evaluate the performance of the low-dimensional data. The accuracy and F1 of the dimension reduction algorithms are recorded in Figure 4. In addition, it should be noted that the number of iterations of the basic 1D CNN and the group CNN algorithms are 1000. The recorded results are the average of 5 experiments.

According to the accuracy and F1 of different dimension reduction algorithms in Figure 4, we can obtain some conclusions. First, for the low-dimensional dataset, such as KDDCUP99, the ratios of the highest accuracy and F1 are high. For the high-dimensional dataset, the ratios of the highest accuracy and F1 are low, such as CICMalDroid2020-470. Furthermore, the highest accuracy and F1 at the low ratios are even higher than that of the original data. Therefore, we think that it is quite necessary to reduce the data dimensions by the dimension reduction algorithms. Second, the accuracy and F1 of different ratios by the group CNN are the highest. Therefore, we can obtain that the group CNN is the best dimension reduction algorithm. At the same time, the accuracy and F1 of the basic 1D CNN are less than that of the

(a) The accuracy and F1 of KDDCUP99 by the dimension reduction algorithms

(b) The accuracy and F1 of CICMalDroid2020-139 by the dimension reduction algorithms

(c) The accuracy and F1 of CICMalDroid2020-470 by the dimension reduction algorithms

FIGURE 4: The accuracy and F1 of the dimension reduction algorithms.

TABLE 4: The structures, FLOP, parameter counts, and running time of the basic 1D CNN and the group CNN.

(a) The structures, FLOP, parameter counts, and running time of the basic 1D CNN

Datasets	Structures of the basic 1D CNN	FLOP	Parameter counts	Running time (S)
KDD99	Layers: 3	$4.63 * 10^9$	45700	602.68
	Layer 1: $C_{in} = 1, C_{out} = 40, M' = 40, W'_1 = 10, W'_2 = 32$			
	Layer 2: $C_{in} = 40, C_{out} = 40, M' = 40, W'_1 = 8, W'_2 = 25$			
CICMalDroid2020-139	Layer 3: $C_{in} = 40, C_{out} = 80, M' = 80, W'_1 = 8, W'_2 = 18$	$6.69 * 10^9$	86798	852.84
	Layers: 4			
	Layer 1: $C_{in} = 1, C_{out} = 30, M' = 30, W'_1 = 20, W'_2 = 128$			
	Layer 2: $C_{in} = 30, C_{out} = 20, M' = 20, W'_1 = 20, W'_2 = 109$			
CICMalDroid2020-470	Layer 3: $C_{in} = 20, C_{out} = 40, M' = 40, W'_1 = 10, W'_2 = 100$	$1480.39 * 10^9$	3768915	48213.86
	Layer 4: $C_{in} = 40, C_{out} = 20, M' = 20, W'_1 = 10, W'_2 = 91$			
	Layers: 5			
	Layer 1: $C_{in} = 1, C_{out} = 249, M' = 249, W'_1 = 10, W'_2 = 461$			
	Layer 2: $C_{in} = 249, C_{out} = 100, M' = 100, W'_1 = 40, W'_2 = 422$			
	Layer 3: $C_{in} = 100, C_{out} = 50, M' = 50, W'_1 = 80, W'_2 = 343$			
	Layer 4: $C_{in} = 50, C_{out} = 20, M' = 20, W'_1 = 40, W'_2 = 304$			
	Layer 5: $C_{in} = 20, C_{out} = 20, M' = 20, W'_1 = 20, W'_2 = 285$			

(b) The structures, FLOP, parameter counts, and running time of the group CNN

Datasets	Structures of the group CNN	FLOP	Parameter counts	Running time (S)
KDD99	Groups: 2; layers: 3	$0.82 * 10^9$	13994	595.77
	Layer 1: $C_{in} = 1, C_{out} = 20, M' = 20, W'_1 = 5, W'_2 = 16$			
	Layer 2: $C_{in} = 20, C_{out} = 20, M' = 20, W'_1 = 4, W'_2 = 13$			
CICMalDroid2020-139	Layer 3: $C_{in} = 20, C_{out} = 40, M' = 40, W'_1 = 4, W'_2 = 10$	$4.90 * 10^9$	61180	484.02
	Groups: 2; layers: 4			
	Layer 1: $C_{in} = 1, C_{out} = 15, M' = 15, W'_1 = 10, W'_2 = 60$			
	Layer 2: $C_{in} = 15, C_{out} = 10, M' = 10, W'_1 = 10, W'_2 = 51$			
CICMalDroid2020-470	Layer 3: $C_{in} = 10, C_{out} = 20, M' = 20, W'_1 = 5, W'_2 = 47$	$23.52 * 10^9$	98976	3237.18
	Layer 4: $C_{in} = 20, C_{out} = 10, M' = 10, W'_1 = 5, W'_2 = 43$			
	Groups: 4; layers: 5			
	Layer 1: $C_{in} = 1, C_{out} = 63, M' = 63, W'_1 = 3, W'_2 = 115$			
	Layer 2: $C_{in} = 63, C_{out} = 25, M' = 25, W'_1 = 10, W'_2 = 106$			
	Layer 3: $C_{in} = 25, C_{out} = 12, M' = 12, W'_1 = 20, W'_2 = 87$			
	Layer 4: $C_{in} = 12, C_{out} = 5, M' = 5, W'_1 = 10, W'_2 = 78$			
	Layer 5: $C_{in} = 5, C_{out} = 5, M' = 5, W'_1 = 5, W'_2 = 74$			

group CNN, but higher than that of PCA and SVD, which are traditional methods. Furthermore, we can conclude that the results of the deep learning methods are better than that of the traditional methods. Therefore, we suggest to apply deep learning algorithms to reduce the dimensions.

4.3.5. Comparison of Running Time. In theory, we have already proved that the parameter counts and FLOP of the group CNN are smaller than that of basic 1D CNN. In this section, we compare the values of FLOP, parameter counts, and running time between the basic 1D CNN and the group CNN. The basic 1D CNN and the group CNN are built with different structures to analyze the datasets. In particular, the numbers of layers and the parameters of each layer are shown in Table 4.

The basic 1D CNN and the group CNN have similar structures, when dealing with the same dataset. It should be noted that the count of convolutional kernels in each layer of the basic 1D CNN is equal to that of the group CNN, which means that the count of convolutional kernels in each layer of the basic 1D CNN is equal to the numbers of the groups multiplied by the counts of convolutional kernels in each group. When the models are operating to analyze the data, running time is recorded. At the same time, FLOP and parameters are computed. The results are shown in Table 4.

Table 4 shows the structures, FLOP, parameters, and running time of the basic 1D CNN and the group CNN. It is easy to find that the more layers of the structures have, the larger the FLOP, parameters, and running time in Table 4(a) and

Table 4(b). Furthermore, the FLOP, parameters, and running time of the group CNN in Table 4(b) are less than that of the basic 1D CNN in Table 4(a), when these two CNN models deal with the same datasets.

In particular, more FLOP, parameter counts, and running time of the group CNN on CICMalDroid2020-470 decrease, compared to that of the group CNN on KDD99 and CICMalDroid2020-139. Maybe, we can infer that the larger the group count is, the more FLOP, parameter counts, and running time reduce. It should be noted that the structures of the basic 1D CNN and the group CNN in this section are set to compare the running time, which are not used in other sections. On the contrary, in other sections, the structures of basic 1D CNN and group CNN are set to obtain the highest performance, which are totally different from that in this section.

5. Conclusions

In this paper, we present a 1D group CNN model to reconstruct the features and reduce the dimensionality. The main characteristic is that grouped data are based on feature correlations, which means that the data are grouped by column. CNN model grouping occurs in convolution kernel grouping. In summary, first, compared to all features, our group CNN can achieve the best performance with fewer features. Second, compared to the basic 1D CNN, the group CNN outperforms the basic 1D CNN on the features at different ratios. Third, compared to the dimension reduction algorithms, the accuracies and F1 of the group CNN are the highest. Fourth, compared to the basic 1D CNN, the FLOP, parameters, and running time of the group CNN are lower. Therefore, from the evaluations of all of the aspects, the group CNN spends less time but achieves better performance with fewer features.

Data Availability

The datasets used to support the findings of this study can be downloaded from the public websites whose references are provided in this paper. And the datasets also are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper is supported by the Natural Science Foundation of Zhejiang Province (Nos. LY20F020012 and LQ19F020008), the National Natural Science Foundation of China (No. 61802094), Zhejiang Electronic Information Product Inspection and Research Institute (Key Laboratory of Information Security of Zhejiang Province), and Key Laboratory of Brain Machine Collaborative Intelligence of Zhejiang Province.

References

- [1] X. Zhou, W. Liang, Z. Luo, and Y. Pan, "Periodic-aware intelligent prediction model for information diffusion in social networks," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 894–904, 2021.
- [2] Z. Cai and Z. He, "Trading private range counting over big IoT data," in *The 39th IEEE International Conference on Distributed Computing Systems*, pp. 144–153, Dallas, TX, USA, 2019.
- [3] L. Qi, C. Hu, X. Zhang et al., "Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 4159–4167, 2020.
- [4] X. Yan, Y. Xu, X. Xing, B. Cui, Z. Guo, and T. Guo, "Trustworthy network anomaly detection based on an adaptive learning rate and momentum in IIoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6182–6192, 2020.
- [5] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial IIoTs," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, 2020.
- [6] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective data-sanitization for preventing sensitive information inference attacks in social networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 577–590, 2018.
- [7] Y. Xu, C. Zhang, G. Wang, Z. Qin, and Q. Zeng, "A blockchain-enabled deduplicatable data auditing mechanism for network storage services," *IEEE Transactions on Emerging Topics in Computing*, 2020.
- [8] Y. Xu, Q. Zeng, G. Wang, C. Zhang, J. Ren, and Y. Zhang, "An efficient privacy-enhanced attribute-based access control mechanism," *Concurrency & Computation Practice & Experience*, vol. 32, no. 5, pp. 1–10, 2020.
- [9] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 766–775, 2020.
- [10] Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen, and Y. Zhang, "Blockchain empowered arbitrable data auditing scheme for network storage as a service," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 289–300, 2020.
- [11] C. Zhang, Y. Xu, Y. Hu, J. Wu, J. Ren, and Y. Zhang, "A blockchain-based multi-cloud storage data auditing scheme to locate faults," *IEEE Transactions on Cloud Computing*, 2021.
- [12] Y. Xu, C. Zhang, Q. Zeng, G. Wang, J. Ren, and Y. Zhang, "Blockchain-enabled accountability mechanism against information leakage in vertical industry services," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1202–1213, 2021.
- [13] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: a survey," *Applied Sciences*, vol. 20, no. 9, p. 4396, 2019.
- [14] J. Cheng, J. Zheng, and X. Yu, "An ensemble framework for interpretable malicious code detection," *International Journal of Intelligent Systems*, 2020.
- [15] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Computing Surveys*, vol. 50, no. 3, pp. 1–40, 2017.
- [16] X. Yan, Y. Xu, B. Cui, S. Zhang, T. Guo, and C. Li, "Learning URL embedding for malicious website detection," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6673–6681, 2020.
- [17] S. M. Ghaffarian and H. R. Shahriari, "Software vulnerability analysis and discovery using machine-learning and data-mining techniques," *ACM Computing Surveys*, vol. 50, no. 4, pp. 1–36, 2017.

- [18] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455–5516, 2019.
- [19] X. Zhou, X. Xu, W. Liang et al., "Intelligent small object detection based on digital twinning for smart manufacturing in industrial CPS," *IEEE Transactions on Industrial Informatics*, 2021.
- [20] C. F. Tsai, Y. F. Hsu, C. Y. Lin, and W. Y. Lin, "Intrusion detection by machine learning: a review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11994–12000, 2009.
- [21] S. Maldonado, R. Weber, and F. Famili, "Feature selection for high-dimensional class-imbalanced data sets using support vector machines," *Information Sciences*, vol. 286, pp. 228–246, 2014.
- [22] R. K. Vigneswaran, R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security," in *2018 9th International Conference on Computing, Communication and Networking Technologies*, Bengaluru, India, 2018.
- [23] S. Ansari, V. Bartos, and B. Lee, "Shallow and deep learning approaches for network intrusion alert prediction," *Procedia Computer Science*, vol. 171, pp. 644–653, 2020.
- [24] X. Zhou, X. Xu, W. Liang, Z. Zeng, and Z. Yan, "Deep learning enhanced multi-target detection for end-edge-cloud surveillance in smart IoT," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12588–12596, 2021.
- [25] Y. Liu, C. Wang, Y. Zhang, and J. Yuan, "Multiscale convolutional CNN model for network intrusion detection," *Computer Engineering and Applications*, vol. 55, no. 3, p. 90, 2019.
- [26] M. Sheikhan and Z. Jadidi, "Flow-based anomaly detection in high-speed links using modified GSA-optimized neural network," *Neural Computing and Applications*, vol. 24, no. 3–4, pp. 599–611, 2014.
- [27] X. Zhou, Y. Li, and W. Liang, "CNN-RNN based intelligent recommendation for online medical pre-diagnosis support," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 3, pp. 912–921, 2021.
- [28] Y. Xu, X. Yan, Y. Wu, Y. Hu, W. Liang, and J. Zhang, "Hierarchical bidirectional RNN for safety-enhanced B5G heterogeneous networks," *IEEE Transactions on Network Science and Engineering*, 2021.
- [29] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, "Generative adversarial networks," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–38, 2021.
- [30] X. Zhou, W. Liang, S. Shimizu, J. Ma, and Q. Jin, "Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5790–5798, 2021.
- [31] X. Yan, B. Cui, Y. Xu, P. Shi, and Z. Wang, "A method of information protection for collaborative deep learning under GAN model attack," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 3, pp. 871–881, 2021.
- [32] 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.
- [33] Q. Yang, W. Shi, J. Chen, and W. Lin, "Deep convolution neural network-based transfer learning method for civil infrastructure crack detection," *Automation in Construction*, vol. 116, no. 10, article 103199, 2020.
- [34] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: a survey," *Mechanical Systems and Signal Processing*, vol. 151, p. 107398, 2021.
- [35] Q. Xiang, X. Wang, Y. Song, L. Lei, R. Li, and J. Lai, "One-dimensional convolutional neural networks for high-resolution range profile recognition via adaptively feature recalibrating and automatically channel pruning," *International Journal of Intelligent Systems*, vol. 36, no. 1, pp. 332–361, 2021.
- [36] S. Chen, J. Yu, and S. Wang, "One-dimensional convolutional auto-encoder-based feature learning for fault diagnosis of multivariate processes," *Journal of Process Control*, vol. 87, pp. 54–67, 2020.
- [37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.
- [38] T. Zhang, G. J. Qi, B. Xiao, and J. Wang, "Interleaved group convolutions for deep neural networks," *ICCV*, vol. 1707, 2017.
- [39] Y. Lu, G. Lu, R. Lin, J. Li, and D. Zhang, "SRGC-nets: sparse repeated group convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 99, pp. 1–14, 2019.
- [40] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.
- [41] M. Kruczkowski and E. N. Szyrkiewicz, "Support vector machine for malware analysis and classification," in *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies*, pp. 415–420, Warsaw, Poland, 2014.
- [42] L. Bilge, S. Sen, and D. Balzarotti, "Exposure: a passive DNS analysis service to detect and report malicious domains," *ACM Transactions on Information and System Security*, vol. 16, no. 4, pp. 1–28, 2013.
- [43] Y. Y. Aung and M. M. Min, "Hybrid intrusion detection system using K-means and classification and regression trees algorithms," in *2018 IEEE 16th International Conference on Software Engineering Research, Management and Applications*, pp. 195–199, Kunming, China, 2018.
- [44] A. Mo, K. Qader, and M. Alkasassbeh, "Comparative analysis of clustering techniques in network traffic faults classification," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 5, no. 4, pp. 6551–6563, 2017.
- [45] D. Abdelhafiz, C. Yang, R. Ammar, and S. Nabavi, "Deep convolutional neural networks for mammography: advances, challenges and applications," *BMC Bioinformatics*, vol. 20, no. 11, pp. 281–301, 2019.
- [46] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42210–42219, 2019.
- [47] Y. Wang, J. An, and W. Huang, "Using CNN-based representation learning method for malicious traffic identification," in *2018 IEEE/ACIS 17th International Conference on Computer and Information Science*, pp. 400–404, Singapore, Singapore, 2018.

- [48] J. Zhang, Z. Qin, H. Yin, L. Ou, and K. Zhang, "A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding," *Computers & Security*, vol. 84, pp. 376–392, 2019.
- [49] J. Zhang, Z. Qin, H. Yin, L. Ou, S. Xiao, and Y. Hu, "Malware variant detection using opcode image recognition with small training sets," in *2016 25th International Conference on Computer Communication and Networks*, Waikoloa, HI, USA, 2016.
- [50] J. Yan, Y. Qi, and Q. Rao, "Detecting malware with an ensemble method based on deep neural network," *Security and Communication Networks*, vol. 2018, 16 pages, 2018.
- [51] C. Ma, X. Du, and L. Cao, "Analysis of multi-types of flow features based on hybrid neural network for improving network anomaly detection," *IEEE Access*, vol. 7, pp. 148363–148380, 2019.
- [52] M. Azizjon, A. Jumabek, and W. Kim, "1D CNN based network intrusion detection with normalization on imbalanced data," in *2020 International Conference on Artificial Intelligence in Information and Communication*, pp. 218–224, Fukuoka, Japan, 2020.
- [53] W. Wei, Q. Ke, J. Nowak, M. Korytkowski, R. Scherer, and M. Woźniak, "Accurate and fast URL phishing detector: a convolutional neural network approach," *Computer Networks*, vol. 178, p. 107275, 2020.
- [54] H. Zhang, L. Huang, C. Q. Wu, and Z. Li, "An effective convolutional neural network based on SMOTE and Gaussian mixture model for intrusion detection in imbalanced dataset," *Computer Networks*, vol. 177, p. 107315, 2020.
- [55] S. Xie and R. Girshick, "Aggregated residual transformations for deep neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, 2016.
- [56] M. D. Zeiler and R. Fergus, *Visualizing and Understanding Convolutional Neural Networks*, Springer International Publishing, 2013.
- [57] Y. Zhang, T. Shen, X. Ji, Y. Zhang, R. Xiong, and Q. Dai, "Residual highway convolutional neural networks for in-loop filtering in HEVC," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3827–3841, 2018.
- [58] M. E. Paoletti, J. M. Haut, X. Tao, J. Plaza, and A. Plaza, "FLOP-reduction through memory allocations within CNN for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 1109, no. 10, pp. 5938–5952, 2020.
- [59] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational LSTM enhanced anomaly detection for industrial big data," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3469–3477, 2021.
- [60] X. Yan, J. Zhang, H. Elahi, M. Jiang, and H. Gao, "A personalized search query generating method for safety-enhanced vehicle-to-people networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5296–5307, 2021.
- [61] S. Hettich and S. D. Bay, *KDD Cup 1999 Data*, The UCI KDD Archive, 1999.
- [62] S. MahdaviFar, A. Kadir, R. Fatemi, D. Alhadidi, and A. A. Ghorbani, "Dynamic android malware category classification using semi-supervised deep learning," in *2020 IEEE Int'l. Conf. on Dependable, Autonomic and Secure Computing, Int'l. Conf. on Pervasive Intelligence and Computing, Int'l. Conf. on Cloud and Big Data Computing, Int'l. Conf. on Cyber Science and Technology Congress (DASC/PiCom/CBDCoM/CyberSciTech)*, pp. 515–522, Calgary, AB, Canada, 2020.