Hindawi
Security and Communication Networks
Volume 2020, Article ID 8850550, 20 pages
https://doi.org/10.1155/2020/8850550

WILEY | Hindawi

Research Article

HYBRID-CNN: An Efficient Scheme for Abnormal Flow Detection in the SDN-Based Smart Grid

Pengpeng Ding, Jinguo Li, Liangliang Wang, Mi Wen, and Yuyao Guan

College of Computer Technology and Science, Shanghai University of Electric Power, Shanghai 200090, China

Correspondence should be addressed to Jinguo Li; lijg@shiep.edu.cn

Received 9 April 2020; Revised 5 July 2020; Accepted 21 July 2020; Published 3 August 2020

Academic Editor: Yin Zhang

Copyright © 2020 Pengpeng Ding 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.

Software-Defined Network (SDN) can improve the performance of the power communication network and better meet the control demand of the Smart Grid for its centralized management. Unfortunately, the SDN controller is vulnerable to many potential network attacks. The accurate detection of abnormal flow is especially important for the security and reliability of the Smart Grid. Prior works were designed based on traditional machine learning methods, such as Support Vector Machine and Naive Bayes. They are simple and shallow feature learning, with low accuracy for large and high-dimensional network flow. Recently, there have been several related works designed based on Long Short-Term Memory (LSTM), and they show excellent ability on network flow analysis. However, these methods cannot get the deep features from network flow, resulting in low accuracy. To address the above problems, we propose a Hybrid Convolutional Neural Network (HYBRID-CNN) method. Specifically, the HYBRID-CNN utilizes a Deep Neural Network (DNN) to effectively memorize global features by one-dimensional (1D) data and utilizes a CNN to generalize local features by two-dimensional (2D) data. Finally, the proposed method is evaluated by experiments on the datasets of UNSW_NB15 and KDDCup 99. The experimental results show that the HYBRID-CNN significantly outperforms existing methods in terms of accuracy and False Positive Rate (FPR), which successfully demonstrates that it can effectively detect abnormal flow in the SDN-based Smart Grid.

1. Introduction

The Smart Grid is a grid system with automatic control and self-protection adjustment capabilities [1]. It is supported by information and communication technology to achieve reliability, security, and real-time requirements [2, 3]. The emerging network architecture Software-Defined Network (SDN) ignores the coaxial hardware structure of the network which separates the control plane and the data plane, and directly implements the virtualized configuration of the switch. It is especially suitable for mobile communication network, wired interconnection network, and sensor network in the Smart Grid [4]. The SDN improves the data transmission capability and network compatibility of the Smart Grid, but it also brings new security issues. The highly centralized network control capability and the damage caused by network abnormal flow intrusion have increased significantly [5]. As the control center of the whole network, the SDN itself may be the target of various attacks, such as DDoS, fake flow, breakthroughs in switches, and attacks on the control layer. The destruction of the SDN will cause all switches under its control to be paralyzed or disorders can have devastated effects on the entire network [6]. In the SDN, collaborative abnormal flow detection across multiple domains requires detailed flow data for each relevant domain, such as the contents of a flow table in the last few seconds. Network abnormal flow has the characteristics of potential and unforeseen attacks. Therefore, the detection technology of network abnormal flow is challenged by the demand for larger-scale and higher-dimensional flow data [7].

Recently, most of these studies are based on state transition [8] and artificial intelligence methods [9]. The method based on state transition requires manual calculation and has low recognition accuracy. The method based on artificial intelligence has more advantages in this respect

because of network big data. However, most of the researches have not carried out in-depth feature learning of network flow. For large-scale network abnormal flow detection, there are mainly two types of methods. The first type of method relies on sampling data, it uses network flow data to establish a library of attack intrusion behavior patterns, and the collected data including the host's system logs or collected from the network nodes matches the established pattern library. If the match is successful, it is proved to be an intrusion; otherwise, it is a normal behavior [10]. This method can effectively identify existing attacks and maintain them effectively and improve network security at the time. However, with the development of computers and the Internet, more and more new types of attacks appear in the field of vision. The detection accuracy of expert systems has fallen sharply. It has been unable to meet the requirements, and the sampling data itself is not accurate, which may cause the loss of useful information.

Another type of method is to utilize machine learning methods to perform feature extraction and detection classification after constructing features. The massive amount of network data makes machine learning methods more effective than judgment methods based on expert systems [11]. The traditional machine learning methods are just a shallow feature learning classifier. They have certain limitations when processing complex data. The feature processing that traditional machine learning must do is time consuming and requires specialized knowledge. The performance of most machine learning algorithms depends on the accuracy of the extracted features. Deep learning reduces the manual design effort of feature extractors for each problem by automatically retrieving advanced features directly from raw data [12]. Previous studies have used deep learning to classify mobile encrypted traffic and achieved excellent results [13, 14]. In [15], the authors investigated several deep learning architectures, including 1D CNN, 2D CNN, LSTM, Stacked Autoencoder (SAE), and Multilayer Perceptron (MLP) for mobile encrypted traffic classification. Based on this, this paper aims to apply the excellent feature learning capabilities of deep learning to the SDN-based Smart Grids to achieve highly accurate network abnormal flow detection.

To meet the above problems and challenges, we hope to apply the excellent feature learning capabilities of deep learning to the SDN-based Smart Grid to achieve highly accurate network abnormal flow detection. The main contributions of this article can be summarized as follows:

(i) First, we design a framework for improving the security of the Smart Grid by applying an abnormal flow detection algorithm in the SDN-based Smart Grid communication network; it can identify abnormal flow and detect the type of attack.

(ii) Second, we propose a deep learning algorithm of Hybrid Convolutional Neural Networks (HYBRID-CNN) to detect abnormal flow in the SDN-based Smart Grid communication network. The HYBRID-CNN adopts dual-channel data input, which can extract effective features from 1D and 2D flow data, use the self-attention mechanism to fuse key features, and finally use the fully connected neural network for detection.

(iii) Third, we compare the proposed method with the single model and verify the performance improvement of the hybrid model. In addition, we discuss a parameter study to optimize the HYBRID-CNN model.

(iv) Fourth, we perform a lot of experimental comparisons on the UNSW_NB15 and KDDCup 99 benchmark dataset. Experimental results show that the HYBRID-CNN significantly outperforms existing approaches in terms of accuracy and False Positive Rate (FPR).

The rest of this article is organized as follows: we discuss related work in Section 2 and introduce the system model and security requirements in Section 3. We then introduce some preliminary knowledge in Section 4. In Section 5, we introduce our proposed algorithm, and then in Section 6 we introduce experimental comparative analysis. Finally, we discuss and conclude in Sections 7 and 8.

2. Related Work

This section discusses two related types of work, namely, traditional machine learning and deep learning. In the SDN-based network controllers, using traditional machine learning and deep learning to develop flexible and efficient abnormal flow detection schemes presents some challenges. One of the main challenges is how to choose an appropriate feature selection method and another challenge is to accurately grasp the correlation between the selected feature and the abnormal flow detection task and the redundancy between these features [16].

2.1. Traditional Machine Learning. Most of the previous studies were based on traditional machine learning methods, such as Support Vector Machine (SVM), Decision Tree, and Naive Bayes. Naive Bayes algorithm is an important algorithm in the field of machine learning and data mining. It is widely used in the field of machine learning classification, such as text classification and medical diagnosis. Ashraf et al. [17] applied Naive Bayes for network intrusion detection; their basic idea is to select the most likely category based on the Bayesian algorithm under the assumption that the classification is based on feature independence. But this method is only simple shallow feature learning, and it has poor performance for large-scale network flow data. Rai et al. [18] used decision tree C4.5 to perform intrusion detection experiments on the NSL-KDD dataset. In this work, 16 attributes were selected as detection features on the dataset. The proposed algorithm can be used for feature-based intrusion detection, but its accuracy is too low, only 79.52%. Reddy et al. [19] proposed a filtering algorithm based on the SVM classifier to perform the classification task on the KDDCup 99 dataset. This method performed well on the training field but performed poorly in the test dataset and

could not effectively detect unknowns' network abnormal flow.

2.2. Deep Learning. In recent years, as a branch of machine learning, deep learning is becoming more and more popular. It is applied to intrusion detection and research shows that deep learning has completely surpassed traditional methods in performance [20]. Kwon et al. [15] utilized Deep Neural Network-based deep learning methods for flow-based anomaly detection. Experimental results evidence that deep learning can be applied to abnormal flow detection in the SDN. Long Short-Term Memory (LSTM) is a special deep learning model of Recurrent Neural Network. It can remember the input and predicted output of any period and solves the problem of gradient vanish and explosion in the Recurrent Neural Network (RNN). LSTM is widely used in the field of Natural Language Processing [21]. Existing researches have been done on abnormal flow detection based on LSTM [22], and they found that the algorithms have a significant performance improvement for sequence learning compared with traditional machine learning methods, but there is still room for improvement in detection rate and accuracy. CNN is a multi-layer network structure learning algorithm. It can learn hierarchical features from a large amount of data and has broad application prospects in the field of abnormal flow detection. Wang et al. [23] proposed an end-to-end classification method for one-dimensional Convolutional Neural Networks. This method integrates feature extraction, feature selection, and classifiers into a unified end-to-end framework and automatically learns original inputs and expectations. The nonlinear relationship between the outputs has obtained good experimental results. However, the one-dimensional data used in this method is not suitable for local feature extraction, resulting in the detection rate less than the ideal one. In [24], the authors present a new technique for network traffic classification based on a combination of RNN and CNN models that can be used for Internet of Things (IoT) traffic, which provides the best detection results. Wang et al. [25] proposed using CNN combined with LSTM to analyze and detect network flow. It utilizes CNN to learn low-level spatial features of network flow for the first time and then uses LSTM to learn high-level temporal features. The Deep Neural Network completes it automatically, and this method has achieved good results in terms of accuracy and detection rate.

Based on the above works, traditional machine learning methods that are typically used in abnormal flow detection often fail and cannot detect many known and new security threats, largely because those approaches provide less focus on accurate feature selection and classification. It is often inefficient for large-scale network flow. For the current deep learning methods like LSTM and CNN, they often pay more attention to the improvement of the model and ignore the original flow structure features. To address the above problems, we propose a HYBRID-CNN deep learning method for more accurate feature learning. The method utilizes two-channel input structure of 1D data and 2D data: using a CNN to extract local features and using a DNN to extract the global features. Specifically, a self-attention mechanism is added to select the most important features.

3. System Model View

In this section, we formalize the system model and system security requirements.

3.1. System Model. The Smart Grid uses two-way communication technology to connect many power components to ensure mutual communication between the components. Implementing the SDN on Smart Grid technology separates network control from data forwarding equipment that includes network infrastructure, thereby enabling logically centralized control and enabling the network to be programmed by a central software unit. The control layer, as the brain of the network, carries the controller software. The software-defined routing rules determine where to route flow. There are programmable network devices in the data plane to route flow according to the rules defined by the controller. The top of the module implements the function of the abnormal flow detection module. As shown in Figure 1, the SDN-based Smart Grid mainly includes the following parts [26].

3.1.1. Physical Plane. This layer is responsible for packet switching and routing. It includes the basic components of network communication in Smart Grid, such as smart meter, Power Management Unit (PMU), various sensors, and various communication equipment. Different from the traditional network, these basic components cannot make decisions independently because of no control unit. They are only responsible for collecting the generated key data and forwarding the collected data to the control layer through the programmable SDN switch infrastructure while complying with the rules defined by the controller.

3.1.2. Southbound Interface. The definition of south interface provides the communication protocol between the physical layer and the control layer. OpenFlow protocol developed by Stanford is currently the most common and standard protocol in south interface [27]. It can realize secure communication in the SDN by determining the message format from a programmable switch to controller.

3.1.3. Control Plane. As the central brain, the control layer has a SDN controller or more whose task is to manage the forwarding behavior of data flow by determining forwarding rules, which need to be written into the flow table of the programmable switch in the physical layer through the south interface.

3.1.4. Northbound Interface. The north interface definition provides an interface for communication between the control layer and the application layer and enables

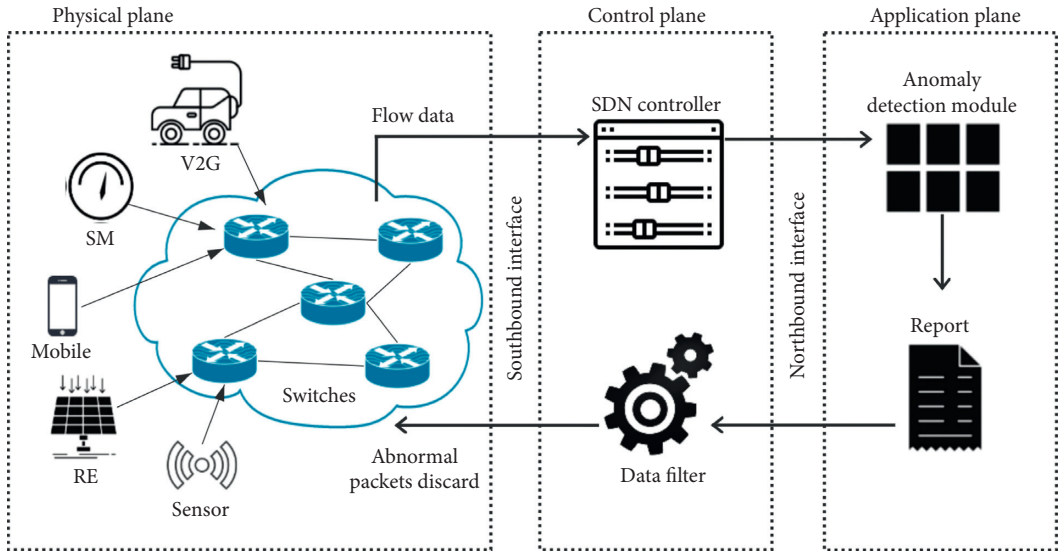

Figure 1: The system model of the SDN-based Smart Grid; it mainly includes physical plane, southbound interface, control plane, northbound interface, and application plane. Devices in the physical layer initiate access request through the Internet, and the flow collection module of the SDN controller captures all request flow statistics table information to extract flow features. HYBRID-CNN is used to detect abnormal traffic and generate abnormal reports. Then, the generated anomaly report is sent to the SDN controller through the security channel. Finally, the SDN controller discards attack packets and updates the flow table according to the received report.

application programs to program the network. It abstracts the details of data in the physical layer and allows network administrators, service providers, and researchers to customize the control rules and behaviors of their networks.

3.1.5. Application Plane. The application layer comprises many Smart Grid applications, including network security function programs such as abnormal flow detection module and flow data filtering module. All these application-defined policies need to be translated into OpenFlow rules that are transferred to the physical layer programmable switch and then transferred from the north interface to the control layer.

3.2. System Security Requirements

3.2.1. The Immovability and Concentricity of Network Architecture. The function of the Smart Grid communication network is generated with the design phase, and it is almost impossible to reconfigure the network based on the real-time needs of the network. In terms of performance and resilience, the bottlenecks will be caused by this nondynamic structure of today's Smart Grid. At the same time, the network will be vulnerable to multiple types of attacks. On the other hand, the highly centralized network control capability increases the damage caused by network abnormal flow intrusion considerably [28]. The SDN is the control center of the entire network. It may itself be the target of various attacks and these attacks will damage the SDN resulting in all its control paralysis or misbehavior of a switch can have a devastating effect on the entire network. Therefore, it is necessary to design an effective abnormal flow detection algorithm in the SDN controller.

3.2.2. The Hierarchy of Network Flow. Network flow has a distinct hierarchy, as shown in Figure 2, where the bottom row shows a sequence of flow bytes. According to a specific network protocol format, multiple flow bytes are combined into a network packet, and then multiple network packets are combined into a network flow. A network flow is divided into normal or malicious tasks, and a deep learning algorithm is used to learn hierarchical features, which has achieved good results. These studies urge us to use deep learning to learn the hierarchical features of network flow to complete the task of intrusion anomaly detection.

3.3. Working Methodology. Devices in the physical layer initiate access request through the Internet, and the flow collection module of the SDN controller captures all request flow statistics table information to extract flow features. The abnormal flow detection module includes three stages: data preprocessing, model training, and model validation, as shown in Figure 3. First, the collected flowmeter data are preprocessed, including data encoding, data normalization, data reshaping, and data split. After data preprocessing, the flow data vectors will be feature-extracted, feature-fused, and anomaly-detection-classified by the HYBRID-CNN algorithm.

In addition to the powerful anomaly flow detection above, the proposed solution performs end-to-end delivery of detection reports through the SDN as shown in Figure 1. This is achieved by incorporating the anomaly flow detection model into the core of the SDN control plane. The execution process works in the following order: (i) detection stage, (ii) reporting phase, and (iii) update phase. In the first stage, the control plane encapsulated with the anomaly flow detection model classifies the incoming flow as abnormal and normal. Then in the second stage, the report is communicated to the

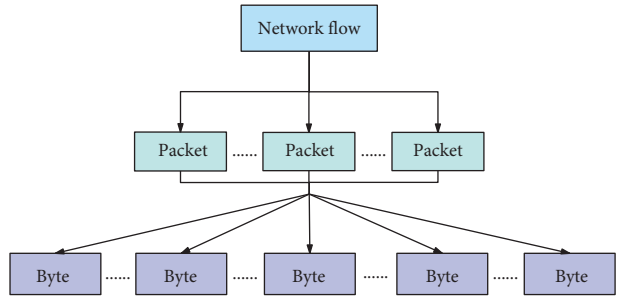

Figure 2: The structure of a network flow. Multiple bytes are combined into a packet, and then multiple packets are combined into a network flow.

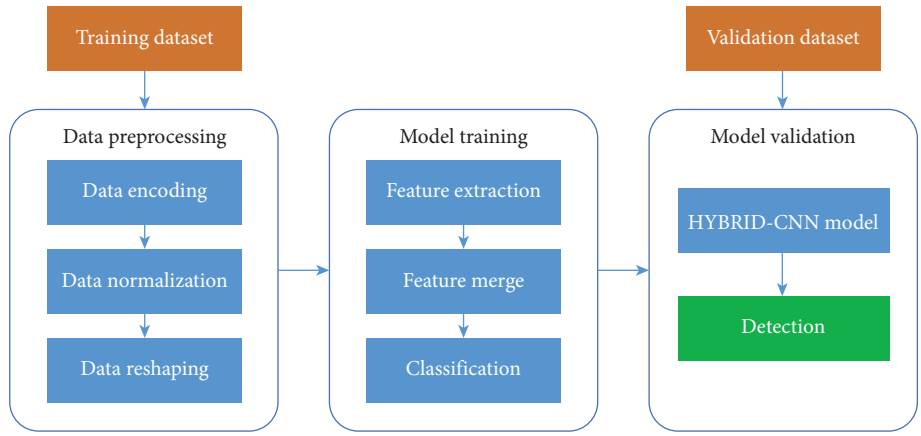

Figure 3: Working methodology of the proposed anomaly detection algorithm; it includes data preprocessing, model training, and model validation.

control plane. If the incoming flow is abnormal, the control plane discards the packet and immediately gives up communication with the requesting host. This helps protect the underlying network with malicious content and prevents it from spreading further on the network. During the update stage, the control plane updates the flow table entry of the forwarding device.

4. Preliminaries

In this section, we briefly describe the general notion used in our proposed algorithm.

4.1. Activation Function. The activation function provides the nonlinear modeling capability of the network. Rectified Linear Unit (ReLU) is the most widely used function [29]; it can keep the gradient from attenuating, thus effectively alleviating the problem of gradient disappearance; the function expression is as follows: the ReLU activation function produces 0 as an output when $x < 0$ and produces a linear with slope of 1 when $x > 0$:

$$\hat{y}' = \max(0, x). \tag{1}$$

4.2. Cross-Entropy Loss. Cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. It increases as the predicted probability diverges from the actual label. In binary classification, where the number of classes **M** equals 2, the cross-entropy loss can be calculated as

$$\text{loss} = -(y\log(p)) - (1 - y)\log(1 - p). \tag{2}$$

If $\mathbf{M} > 2$ (i.e., multiclass classification), we calculate a separate loss for each class label per observation and sum the results:

$$\text{loss}' = -\sum_{c=1}^{M} y_{o,c}\log\left(p_{o,c,}\right), \tag{3}$$

where y is binary indicator (0 or 1) if class label c is the correct classification for observation o and p is predicted probability that observation o is of class c.

4.3. Optimizer. We use Adam optimizer to learn the network weight parameters. And independent adaptive learning rates are designed for different parameters with calculating the first-order moment estimation and the second-order moment estimation of the gradient. Empirical results prove that Adam has greater advantages over other optimizers in practice [30]. Moving averages of gradient $m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t$ and squared gradient $v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$, bias corrected estimators for the

first moments $\hat{m}_t$ and second moments $\hat{v}_t = v_t/(1-\beta_2^t)$, the update rules for Adam are as follows:

$$\omega_t = \omega_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}, \tag{4}$$

where ω is model weights, η is the step size, and β, ε are hyperparameters.

5. Proposed HYBRID-CNN Algorithm

In this part, we first introduce the data preprocessing operation. Then, we describe the structure of HYBRID-CNN algorithm and how to detect abnormal flow.

5.1. Data Preprocessing

5.1.1. Data Encoding. The input flow data contains a variety of features; some of them are no-numeric types, so they need to be encoded as numeric types to be used as input to the neural network. Here, we use Label encoder encoding to convert discrete features to continuous features [31], such as [protocol: TCP, service: HTTP, state: FIN, ...] ⟶ [protocol: 4, service: 2, state: 2, ...].

5.1.2. Data Normalization. Data normalization can speed up the solution, improve the accuracy of the model, and prevent a feature with a particularly large value range from affecting the distance calculation. For the features that there is a very large scope in the difference between the minimum and maximum values, such as "dur," "sbytes," and "dbytes," we apply the logarithmic scaling method for scaling to obtain the features which are mapped to a range. We choose the MIN-MAX scaling method [31] and normalize the data according to the following equation:

$$X_i = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}}, \tag{5}$$

where X_i denotes each data point, $X_{\min}$ denotes the minimum value from all data points, and $X_{\max}$ denotes the maximum value from all data points for each feature.

5.1.3. Data Reshaping. For CNN input, its format should be three-dimensional data (height, width, channel), and as a single sample, the channel should be 1, so that we can reshape a single flow sample with a length of $s = h^* w + 1$ to obtain a data structure similar to an image and construct a matrix M of $h^* w$, namely,

$$M' = \begin{pmatrix} M_{11} & \cdots & M_{1w} \\ \vdots & \ddots & \vdots \\ M_{h1} & \cdots & M_{hw} \end{pmatrix}. \tag{6}$$

5.1.4. Data Split. For every model we want to train, each model has two datasets: one is the training dataset and the other is the validation dataset. As shown in Figure 4, in order to separate them, we first apply the shuffle method on the dataset to generate random data and then slice the entire dataset to obtain a training dataset and a validation dataset.

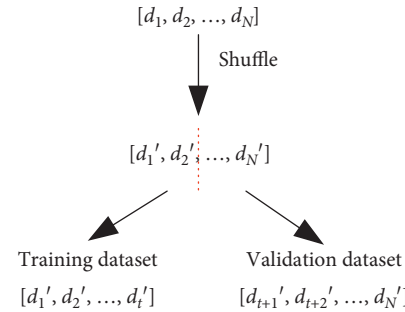

Figure 4: Data split. Using the shuffle method on the network flow dataset to generate random data, and then splitting the random data into a training dataset and a validation dataset.

5.2. HYBRID-CNN.

The structure of CNN is shown in Figure 5. It is an end-to-end deep learning model with powerful feature learning and classification capabilities. It is widely used in image classification, speech recognition, computer vision, and other fields [32].

The network flow contains both abnormal and normal flow, and HYBRID-CNN training is performed at this stage to detect misused attacks, which aims to further categorize the malicious data from stages into corresponding classification strategies, i.e., Scan, R2L, DoS, and Probe. The structure of our proposed HYBRID-CNN algorithm is shown in Figure 6. We divide it into three parts. The first part is feature extraction, the second part is feature fusion, and the third part is the detection classification.

5.2.1. Feature Extraction. In the feature extraction phase, we use the form of dual input of flow data, which aims to extract the features of flow more comprehensively. The role of the input layer is to receive input data, and the size of the input layer is consistent with the size of the input data, such as a vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, or a matrix $\mathbf{M}$.

For the first input (the upper part of the blue box), every user's access flow essentially is 1D data. We utilize two layers of DNN to extract the global features of the flow. Our motivation is to learn the frequent co-occurrence of features pass by memorizing one-dimensional data. The calculation method of each neuron in the fully connected layer is

$$x_i = f\left(\sum_{i=1}^{n} \omega_{i,j} x_i + b_1\right). \tag{7}$$

After the data preprocessing, its input shape is $(h^* w, 1)$. In layer 1, we set a neuron, and the shape of the output data is $(h^* w, a)$. In the fully connected layer 2, we set b neurons, and the shape of the output data is $(h^* w, b)$. The two-dimensional data is straightened to obtain a one-dimensional feature vector of $h^* w^* b, 1$. In this process, the activation function used is ReLU to obtain the output feature O_{wide}.

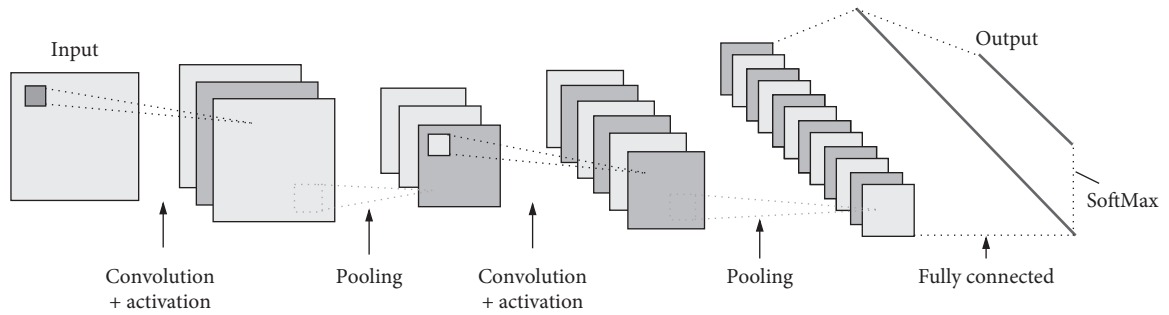

Figure 5: The structure of a CNN. It includes input layer, convolution layer, activation function, pooling layer, fully connected layer, and output layer.

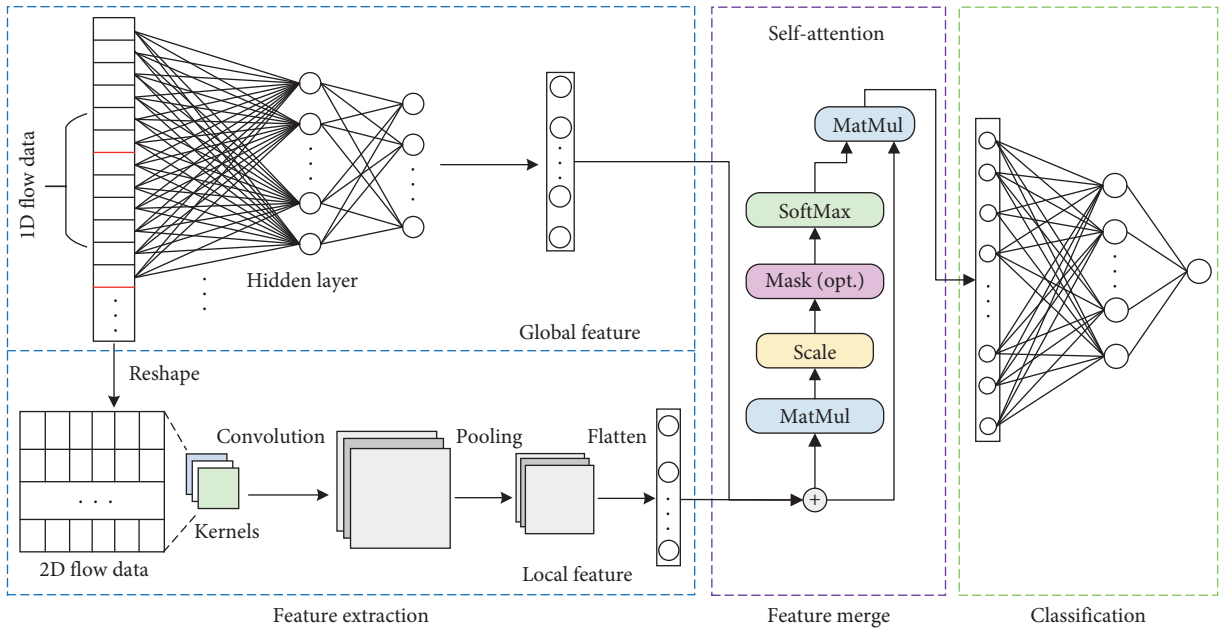

Figure 6: The structure of the proposed HYBRID-CNN algorithm; it includes feature extraction, feature merge, and classification. The feature extraction aims to extract the feature of flow more comprehensively, the self-attention mechanism aims to fuse key feature, and the classification aims to classify accurately.

For the second input (the lower part of the blue box), we reshape the one-dimensional data of the first input into a two-dimensional matrix. We believe that the deeper features can be better learned in the form of two-dimensional matrix input. The CNN uses a sliding convolution kernel to extract local features of flow data. In this part of the network, a convolution layer, a pooling layer, and a flatten layer are included.

One of the limitations of conventional neural networks is poor scalability due to the full connection of neurons; CNN overcomes this shortcoming by convolving each neuron to its neighbors instead of all neurons [33]. Set the input of the i-th layer to x^{l+1}, the output to x^l, and the convolution kernel to k. The convolution operation is performed by the following equation:

$$x^l = f\left(\sum x_i^{l-1} \otimes k_i^l + b^l\right), \tag{8}$$

where $f(\cdot)$ is a nonlinear activation function, $\otimes$ is a convolution sign, and b^l is a bias term. The pooling layer is usually placed after the convolutional layer. By performing a merge operation on a local area of the feature map, the feature has a certain spatial invariance. The merge operation reduces feature size and prevents overfitting. x^{l+1} is obtained by the following pooling:

$$x^{l+1} = \beta\ \mathrm{down}\left(x^l\right) + b, \tag{9}$$

where $\mathrm{down}(\cdot)$ represents the pooling function, β is a multiplicative bias, and b is additive bias. The reshaped shape of the input data is (h, w). We use k convolution kernels with the same shape to extract the convolution features. At first, the data shape is $(h - k + 1, k)$; after pooling, the shape of the data is $((h - k + 1)/2, k)$. Then, through the flatten layer, the data shape is $((h - k + 1)/2^*k, 1)$, and the output feature O_{CNN} is obtained.

For the two extracted features, perform feature fusion to obtain the feature $O_i(k)$:

$$O_i(k) = O_{\mathrm{wide}} + O_{\mathrm{CNN}}. \tag{10}$$

5.2.2. Feature Merge. In the feature fusion part, we use a self-attention mechanism to fuse key features. The essence of the self-attention mechanism is to observe a specific part according to the observation of the need [34].

For self-attention, we get three matrices Q (Query), K (Key), and V (Value) from the input $O_i(k)$. The self-attention mechanism obtains different representations, calculates scaled dot-product attention of each representation, and finally concatenates the results. Specifically, the current representations input into the self-attention layer, and the new representation is calculated. First, we have to calculate the point product between Q and K, and then in order to prevent the result from being too large, it will be divided by a scale $\sqrt{d_k}$, where d_k is the dimension of a query and key vector, and then the results are normalized to a probability distribution using a SoftMax operation and then multiplied by the matrix V to obtain a weighted summation representation. This operation can be expressed as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{11}$$

5.2.3. Classification. After feature fusion, we use a fully connected layer for detection and classification; all neurons in the previous layer are connected to each neuron in the current layer. The fully connected layer is located before the output layer. After the extracted features are converted into a one-dimensional feature vector, they are connected to each neuron in the current layer to map the high-level features in a targeted manner:

$$x_i' = f\left(\sum_{i=1}^{n} \omega_{i,j} x_i + b_1\right). \tag{12}$$

The fully connected layer will target high-level features according to the specific tasks of the output layer perform mapping and use the SoftMax and Sigmoid activation function after mapping to get the final classification detection result (normal, abnormal, or attack types).

The output layer is a SoftMax function [35]; it normalizes K real numbers into a K probabilities distribution, after applying SoftMax, each component will be in the interval $(0, 1)$, and the components will add up to 1, which can be interpreted to map the nonnormalized output of a network to a probability distribution over predicted output classes. Set $z = (z_1, \ldots, z_K) \in \mathbb{R}^K$; the standard SoftMax function σ: $\mathbb{R}^K \longrightarrow \mathbb{R}^K$ is defined by the formula:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \quad \text{for } j = 1, \ldots, K. \tag{13}$$

Hence, the predicted class would be $\hat{y}$:

$$\hat{y} = \arg\max\left[\sigma(z)_j\right]. \tag{14}$$

6. Experimental Evaluation

To evaluate the proposed abnormal flow detection scheme, we conduct the simulation on a 64-bit computer with Intel (R) i7-9750 Hz 2.60 GHz CPU, 8 GB RAM, NVIDIA GeForce RTX 2060 6G GDDR6 GPU, and 10.2 CUDA, using Python, Scikit-learn, NumPy, Pandas, TensorFlow, and Keras. The data we use comes from an online public dataset. We carried out model comparison experiments to verify that the mixed model has higher accuracy than the single model. Compared with traditional machine learning methods and deep learning methods, the experimental results show that our method is superior to these methods.

6.1. Experimental Setup

6.1.1. Experimental Data. The dataset we are using is UNSW_NB15 on network intrusion detection [36], which is a mixture of real normal activity flow and attack flow created by the Australian Network Security Center in the network laboratory using IXIA Perfect Storm tool. Table 1 is the list of features and categories.

These features are categorized into five groups:

(i) Basic features: they involve the attributes that represent protocols connections

(ii) Flow features: they include the identifier attributes between hosts (e.g., server-to-client or client-to-serve)

(iii) Content features: they encapsulate the attributes of TCP/IP; also, they contain some attributes of http services

(iv) Time features: they contain the attributes time, for example, arrival time between packets, start/end packet time, and round-trip time of TCP protocol

(v) Additional generated features: this category can be further divided into two groups: general-purpose features, whereby each of them has its own purpose, to protect the service of protocols, and connection features that are built from the flow of 100 record connections based on the sequential order of the last time feature

To label this dataset, two attributes were provided: attack_cat represents the nine categories of the attack and the normal, and label is 0 for normal and otherwise is 1.

6.1.2. Performance Metrics. The performance metrics for abnormal flow detection depend on the confusion matrix constructed for any proven classification problem [37]. Its size depends on the number of classes contained in the dataset. Its main purpose is to compare the actual tags with the predicted tags. The intrusion detection problem can be defined by a 2×2 confusion matrix, which includes normal and attack categories for evaluation. The detailed description of the confusion matrix is shown in Table 2.

TP and TN denote the conditions for correct classification, while FP and FN denote the conditions for the mistaken classification. TP and TN refer to correctly classified attack flow and normal flow, respectively, while

Table 1: Features of the UNSW_NB15 dataset.

No.	Feature name	Category
(1)	dur	Numeric
(2)	Proto	Nonnumeric
(3)	service	Nonnumeric
(4)	state	No-numeric
(5)	spkts	Numeric
(6)	dpkts	Numeric
(7)	sbytes	Numeric
(8)	dbytes	Numeric
(9)	rate	Numeric
(10)	sttl	Numeric
(11)	dttl	Numeric
(12)	sload	Numeric
(13)	dload	Numeric
(14)	sloss	Numeric
(15)	dloss	Numeric
(16)	sinpakt	Numeric
(17)	dinpakt	Numeric
(18)	sjit	Numeric
(19)	djit	Numeric
(20)	swin	Numeric
(21)	dwin	Numeric
(22)	stcpb	Numeric
(23)	dtcpb	Numeric
(24)	tcprtt	Numeric
(25)	synack	Numeric
(26)	ackdat	Numeric
(27)	smean	Numeric
(28)	dmean	Numeric
(29)	trans_depth	Numeric
(30)	response_body_len	Numeric
(31)	ct_srv_src	Numeric
(32)	ct_state_ttl	Numeric
(33)	ct_dst_ltm	Numeric
(34)	ct_src_dport_ltm	Numeric
(35)	ct_dst_sport_ltm	Numeric
(36)	ct_dst_src_ltm	Numeric
(37)	is_ftp_login	Numeric
(38)	ct_ftp_cmd	Numeric
(39)	ct_flw_http_mthd	Numeric
(40)	ct_src_ltm	Numeric
(41)	ct_srv_dst	Numeric
(42)	is_sm_ips_ports	Numeric

Table 2: Confusion matrix for binary classification problem.

Predicted	Actual	
	Negative	Positive
Negative	TN (true negative)	FP (false positive)
Positive	FN (false negative)	TP (true positive)

FP and FN refer to misclassified normal and attack records, respectively. These four items are used to generate the following performance evaluation metrics.

The Accuracy (Acc) is a measure used to evaluate the overall success rate of the model in detecting normal records and abnormal flow and is calculated as

$$\text{Acc} = \frac{\text{TN} + \text{TP}}{\text{TP} + \text{FP} + \text{TN} + \text{TP}}. \quad (15)$$

The Detection Rate (DR), also known as the True Positive Rate (TPR), is the ratio of correctly classified malicious flow instances to the total number of malicious flow instances. The calculation formula is

$$\text{DR} = \frac{\text{TP}}{\text{FN} + \text{TP}}. \quad (16)$$

The False Positive Rate (FPR) is the proportion of normal instances that are misclassified as attack flow in the total number of normal instances. The formula is

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \quad (17)$$

The Precision (Pre) represents the proportion of the actual normal samples to the samples divided into normal; the formula is

$$\text{Pre} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \quad (18)$$

The $F1$ score is used to synthesize precision and recall as an evaluation index. The formula is

$$F1\text{score} = \frac{2^{*}\text{Pre}^{*}\text{DR}}{\text{Pre} + \text{DR}}. \quad (19)$$

6.2. Performance Comparison

6.2.1. Model Comparison. For comparison, we used a single CNN model and a simple DNN model. Our proposed hybrid CNN model includes 2 input layers, 1 convolutional layer, 1 pooling layer, and 4 fully connected layers. A single CNN model includes a convolutional layer, a pooling layer, and a fully connected layer. The simple DNN model contains only 3 fully connected layers.

The configuration of the model structure parameters in this paper is shown in Figure 7. Each column is a model. The input data shape of the DNN part of our proposed hybrid CNN model is (42,1), the data shape through Dense1 is (42,128), the data shape through Dense_2 is (42,64), and then the data shape through Flatten_1 is (2688), the shape of the input data of the CNN is (6,7) through the Conv1D_1 layer, the shape of the data becomes (4,32), followed by Pooling_1, and the shape of the data becomes (2,32). In the Merge layer, the two-channel data are merged into one. After this layer, the shape of the data becomes (2752) and then passes through the Dense_3 layer. As a result, the same shape is formed in each model by these layers in turn.

As shown in Table 3, we set the initial weight parameters to random values, set the batch size to 512, and use our Adam optimizer and binary_cross-entropy loss function to compile the model. To evaluate the performance of the model, we use accuracy as a metric function during training verification.

After the model is compiled, we use the input data to perform model training in batch mode and evaluate the performance index values at the end of each epoch. One epoch means that all training datasets have undergone a complete training iteration. The training results are shown in Figure 8, where the horizontal axis represents the number of

<table>
<tr><td colspan="5">Hybrid CNN model</td><td colspan="2">Single DNN model</td><td colspan="2">Single CNN model</td></tr>
<tr><td colspan="2">Input1: (42, 1)</td><td colspan="2">Input2: (6, 7)</td><td></td><td colspan="2">Input4: (42, 1)</td><td colspan="2">Input3: (6, 7)</td></tr>
<tr><td rowspan="2">Dense_1</td><td>(42, 1)</td><td rowspan="2">Conv1D</td><td>(6, 7)</td><td></td><td rowspan="2">Dense_1</td><td>(42, 1)</td><td rowspan="2">Conv1D</td><td>(6, 7)</td></tr>
<tr><td>(42, 128)</td><td>(4, 32)</td><td></td><td>(42, 128)</td><td>(4, 32)</td></tr>
<tr><td rowspan="2">Dense_2</td><td>(42, 128)</td><td rowspan="2">Pooling</td><td>(4, 32)</td><td></td><td rowspan="2">Dense_2</td><td>(42, 128)</td><td rowspan="2">Pooling</td><td>(4, 32)</td></tr>
<tr><td>(42, 128)</td><td>(2, 32)</td><td></td><td>(42, 128)</td><td>(2, 32)</td></tr>
<tr><td rowspan="2">Flatten_1</td><td>(42, 64)</td><td rowspan="2">Flatten_2</td><td>(2, 32)</td><td></td><td rowspan="2">Flatten_1</td><td>(42, 64)</td><td rowspan="2">Flatten_2</td><td>(2, 32)</td></tr>
<tr><td>2688</td><td>64</td><td></td><td>2688</td><td>64</td></tr>
<tr><td colspan="5">Attention merge layer: (2752)</td><td colspan="2">–</td><td colspan="2">–</td></tr>
<tr><td colspan="9">Dense_3: (32)</td></tr>
<tr><td colspan="9">Dense_4: (1)</td></tr>
</table>

Figure 7: Model configuration parameters.

Table 3: Configuration parameters for different models.

Methods	Origin weights	Batch_size	Activation
Single DNN model	Random	512	ReLU
Single CNN model	Random	512	ReLU
Our proposed model	Random	512	ReLU

epochs trained, and the vertical axis represents the loss and accuracy score values. We observe that the loss of our proposed hybrid CNN model becomes smaller and smaller as the training progresses, and after 100 epochs of training, it obtains higher accuracy scores than the single CNN model and DNN model.

6.2.2. Method Comparison. To evaluate the performance of our proposed hybrid CNN model, we performed experiments on UNSW_NB15 dataset. The comparison methods selected are as follows:

(i) Naive Bayes [17]: Naive Bayes is a supervised learning classifier based on Bayes theorem. It classifies the problem by combining previous calculated likelihood and probabilities to make the next probability using Bayes rule.

(ii) SVM [19]: an SVM is a discriminative classifier formally defined by separating hyperplanes. SVM-based kernels classify the data which effectively works for most of the datasets. Discriminant function: "Linear SVM."

(iii) LSTM [22]: the improved model based on RNN for intrusion detection, using ReLU activation function, Adam optimizer, 100 epoch, and two-layer LSTM {128, 64}.

(iv) CNN-LSTM [25]: a CNN combined with LSTM to analyze and detect network flow. It utilizes CNN to learn low-level spatial features of network flow for the first time and then uses LSTM to learn high-level temporal features, using ReLU activation function, Adam optimizer, 100 epoch, and two-layer LSTM {128, 64}; two-layer CNN includes pooling layer.

Table 4 lists the performance comparison between our proposed HYBRID-CNN and some other existing methods. It is worth noting that we select a subset for experiments based on a certain training dataset ratio. The training dataset ratio is defined as the proportion of training samples. The proportion of the dataset is 60%, 70%, and 80%. In each dataset of experiments, we evaluated five methods including our proposed method and evaluated three performance metrics (Acc, DR, FPR). The experimental results in Table 4 show that our proposed HYBRID-CNN compared with other traditional machine learning methods and deep learning methods. Compared with other methods, our proposed HYBRID-CNN can reach Accuracy of 0.9564, DR of 0.9856, and FPR of 0.0442, which means that our proposed method has higher accuracy in detecting abnormal flow than other traditional methods. It is because the combination input using a DNN and CNN has better feature learning capabilities.

Figure 9 is a comparison of the training and validation accuracy and loss between our proposed HYBRID-CNN method and the other two methods. All models have been trained for 100 epochs, and performance indicators have been evaluated after each epoch. By comparison, we can find HYBRID-CNN in the training and validation process of the method; the loss convergence speed is much faster. And the best results can be achieved faster for the accuracy improvement, which is obviously better than other methods.

6.2.3. ROC Curves Comparison. We further plot the Receiver Operating Characteristic (ROC) curves of our proposed HYBRID-CNN and state-of-the-art methods on UNSW_NB15, as shown in Figure 10. The ROC curve of HYBRID-CNN is the closest one to the upper left corner,

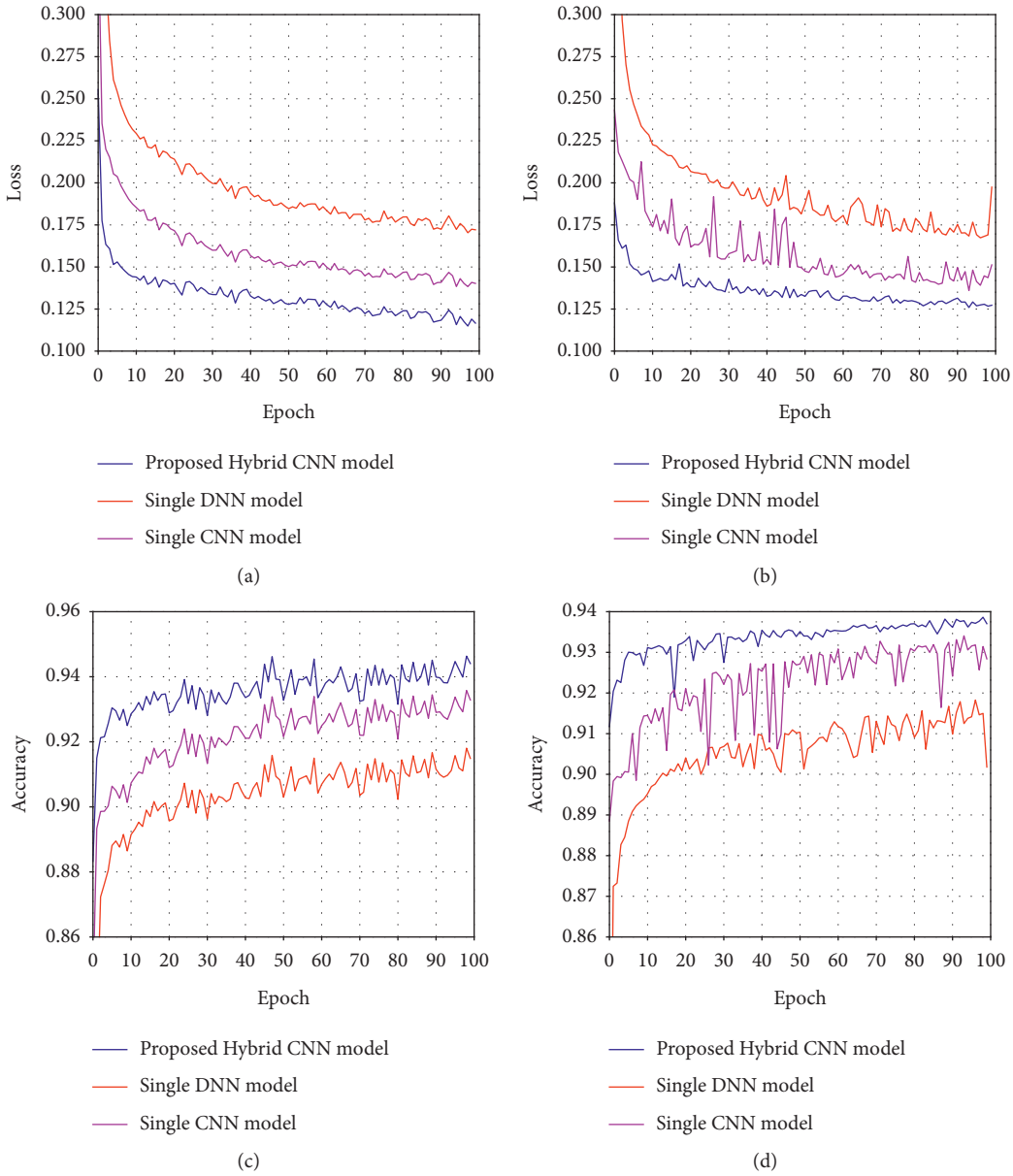

Figure 8: Comparison of different models. (a) Training loss. (b) Validation loss. (c) Training accuracy. (d) Validation accuracy.

Table 4: Performance comparison of the proposed and state-of-the-art methods.

Reference	Method	Proportion = 80%			Proportion = 70%			Proportion = 60%		
		Acc	DR	FPR	Acc	DR	FPR	Acc	DR	FPR
Ashraf et al. [17]	Naive Bayes	0.7663	0.8514	0.3841	0.7669	0.8611	0.3999	0.7655	0.8512	0.3883
Reddy et al. [19]	SVM	0.7594	0.6895	0.1170	0.7257	0.7806	0.3714	0.7346	0.7874	0.3591
Xin et al. [22]	LSTM	0.8916	0.9843	0.2724	0.8897	0.9840	0.2775	0.8894	0.9835	0.2778
Wang et al. [25]	CNN-LSTM	0.8995	0.9612	0.2095	0.8965	0.9460	0.1910	0.8955	0.9571	0.2138
Proposed method	**HYBRID-CNN**	**0.9564**	**0.9856**	**0.0442**	**0.9408**	**0.9382**	**0.0544**	**0.9386**	**0.9493**	**0.0803**

indicating better generalization ability against the other methods. All the results reported above demonstrate that HYBRID-CNN outperforms its competitors. We can conclude that HYBRID-CNN effectively handles the abnormal flow detection problem by the ability to compress the original data to more discriminative abstract features, and HYBRID-CNN is capable of efficient abnormal flow detection.

6.2.4. Computation Comparison. To deepen this investigation, Table 5 reports the number of training parameters

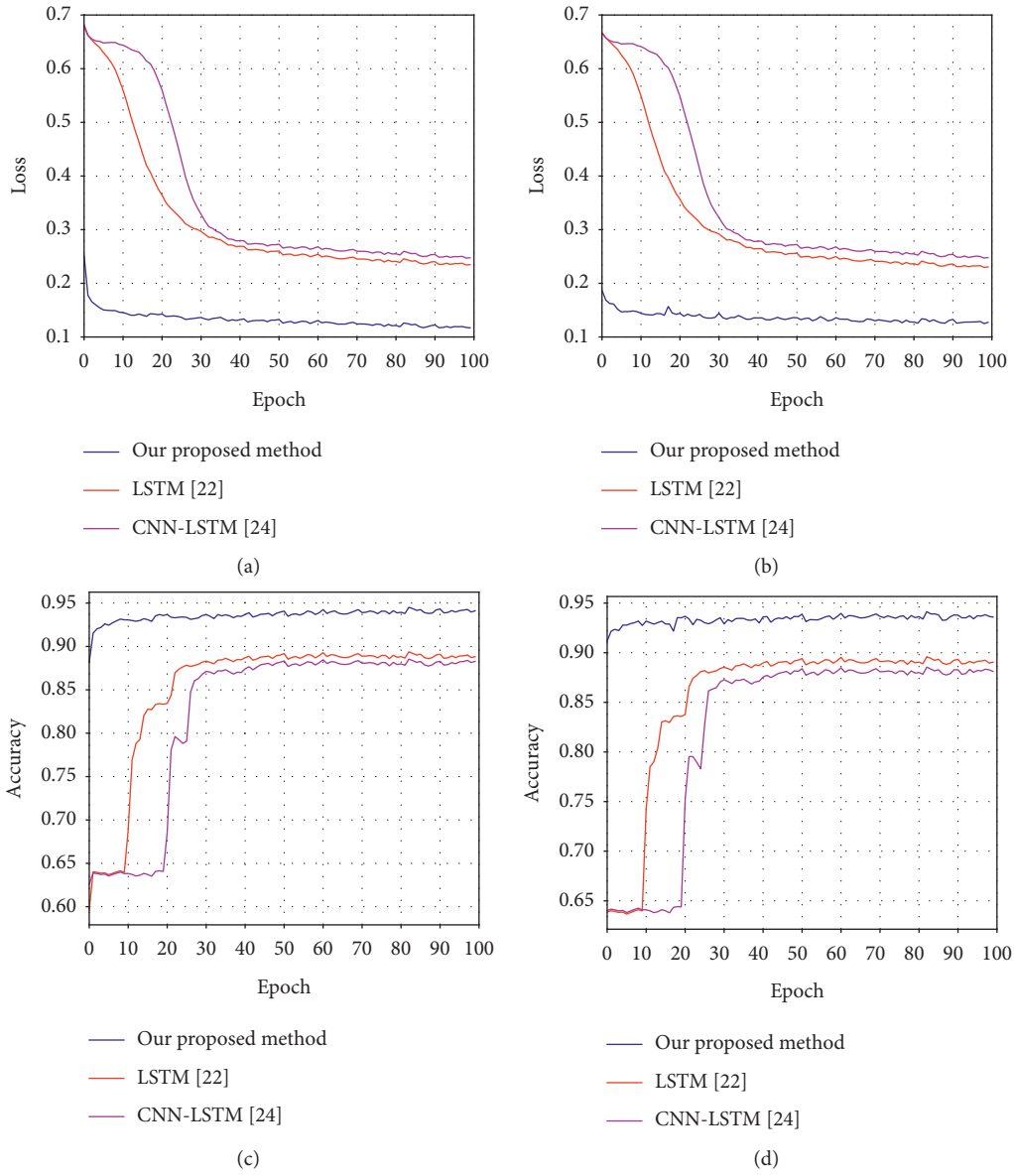

Figure 9: Comparison of different methods. (a) Training loss. (b) Validation loss. (c) Training accuracy. (d) Validation accuracy.

(in millions) and running time required for both the proposed HYBRID-CNN and state-of-the-art methods. We use GPU to accelerate the training speed of all models. It can be noticed that, when training on the UNSW_NB15 dataset, the proposed HYBRID-CNN has fewer trainable parameters and lower training time and testing time. This outcome results from the use of CNN in the proposed method, which can realize efficient parallel computation, and we use as small number of parameters as possible in the structure.

6.3. Parameter Study. There are various configurable hyperparameters in the model, such as Batch_size α, number of convolution kernels β, convolution kernel size γ, and optimizer ε. These hyperparameters can only be configured manually but cannot be optimized automatically through the training process, which will greatly affect the performance of the model. Batch_size α is the number of training samples of the neural network after one forward-propagation and back-propagation operation, which means how many samples will be used to evaluate the loss in each optimization process; β is the number of different convolution kernels used in convolution operation, how many convolution kernels there are, and how many feature maps will be generated after convolution; γ is the size of convolution kernels. Each convolution kernel has three dimensions of length, width, and depth. In a convolution layer of CNN, the length and width of convolution kernels need to be manually configured. Optimizer ϵ is the type of optimizer used to optimize loss and then update weight parameters. Therefore, we deeply analyzed the influence of these super parameters on the performance of our proposed hybrid

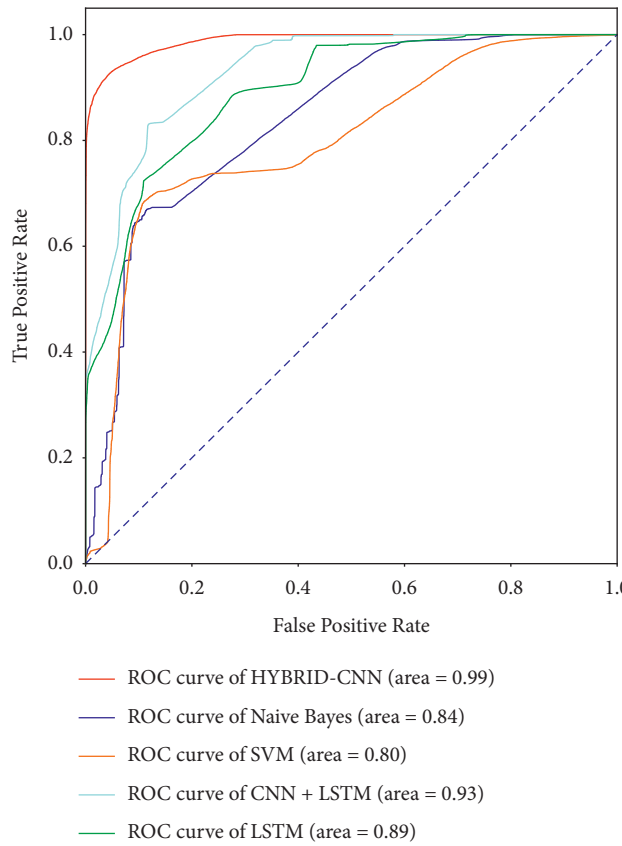

Figure 10: ROC curves of HYBRID-CNN and state-of-the-art methods on UNSW_NB15 dataset.

Table 5: The comparison of the computational complexity of the proposed and state-of-the-art methods.

Method	Trainable parameters	Training time (s)	Testing time (s)
LSTM	0.1391	402.58	1.77
CNN-LSTM	0.1404	526.31	8.99
Proposed	0.0951	271.26	0.75

CNN model. In Figure 7, the parameters of the hybrid CNN model proposed by us are $\alpha = 512$, $\beta = 4$, $\gamma = 1 \times 3$, and $\varepsilon =$ Adam. The model training results for these parameters are as follows.

6.3.1. Effect of Batch_size α. As shown in Figure 11, we set α to 128, 256, and 512 for experiments. When $\alpha = 128$, the training and validation loss converge faster in the same period and finally reach the set number of iterations. The best effect is 0.9477. We can know that a smaller Batch_size can speed up the optimization in the same period, but it means that more calculation time is needed to optimize. Increasing the Batch_size properly can improve the running speed and gradient descent direction. With accuracy increasing, the amplitude of training vibration decreases.

6.3.2. Effect of Number of Convolution Kernels β. As shown in Figure 12, we set the number of convolution kernels β as 1, 2, and 4 for experiments. When the number of convolution kernels is 1, we can get an accuracy of 0.9403. When the number of convolution kernels increases to 2, the loss convergence rate also increases. At 4, the speed of loss convergence is significantly accelerated. Generally, when the network is deeper, more convolution kernels are often required to fully extract key features.

6.3.3. Effect of Convolution Kernel Size γ. As shown in Figure 13, we set the size γ of the convolution kernel to 1×2, 1×3, and 1×4 for experiments. When the size of the convolution kernel is 1×2, the training loss and accuracy rate will jitter sharply. It is not conducive to convergence. When the size of the convolution kernel is increasing, the loss converges a little faster and the fluctuation range becomes smaller, so it should be better to choose a 1×3 or 1×4 size convolution kernel.

6.3.4. Effect of Optimizer ε. As shown in Figure 14, we have selected several commonly used optimizers SGD, RMSprop, Adam, and Adagrad for experimental comparison. When

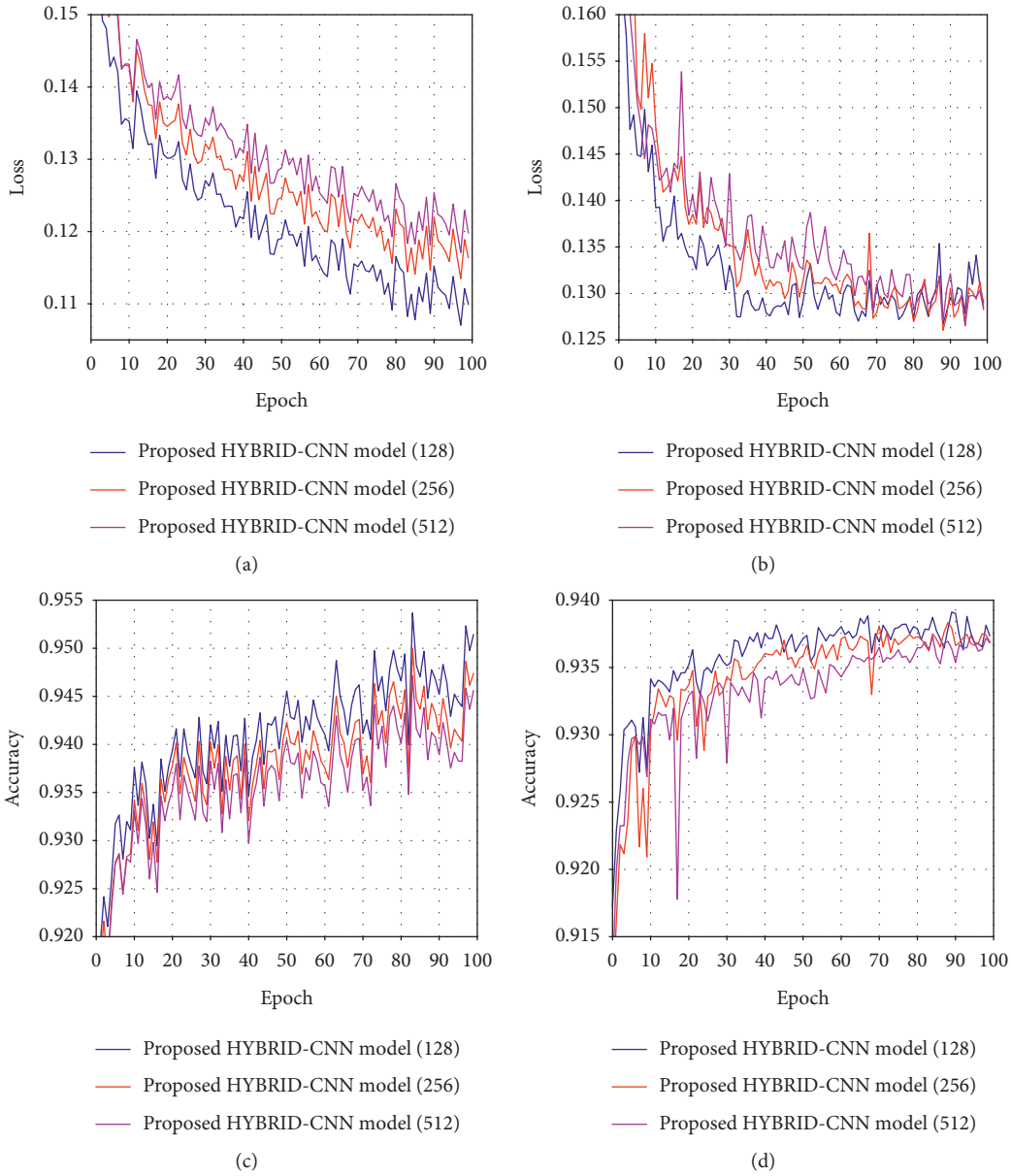

Figure 11: Parameter study of α. (a) Training loss. (b) Validation loss. (c) Training accuracy. (d) Validation accuracy.

SGD is used as an optimizer, the effect is not ideal. It can only achieve an accuracy of 0.9259. There was a large shock at around 40. We can see that when Adam optimizer is used, the initial loss convergence is like other optimizers. In the medium term, the Adam optimizer loss convergence is significantly faster and finally achieves the best. The accuracy is 0.9483.

6.4. Ablation Study. For a thorough analysis, we conduct an ablation study on HYBRID-CNN to analyze the effectiveness of each module. The details of the ablation study based on UNSW_NB15 are listed as follows:

(1) w/o attention: we remove the self-attention module from HYBRID-CNN but keep the DNN module and the CNN module

(2) w/o DNN: the DNN module is removed from HYBRID-CNN

(3) w/o CNN: the CNN module is removed from HYBRID-CNN

We further analyzed the detailed performance of HYBRID-CNN in the ablation study, and the results of the ablation studies are shown in Table 6. Comparing HYBRID-CNN with model (1), we can conclude that the self-attention module can help detect abnormal flow, because attention can capture key features more comprehensively. The effectiveness of DNN can also be demonstrated by comparing HYBRID-CNN with model (2). When we removed the DNN module, accuracy declined because the model could not extract high-dimensional global features. However, when the CNN module was removed, it could be found that the accuracy was greatly reduced,

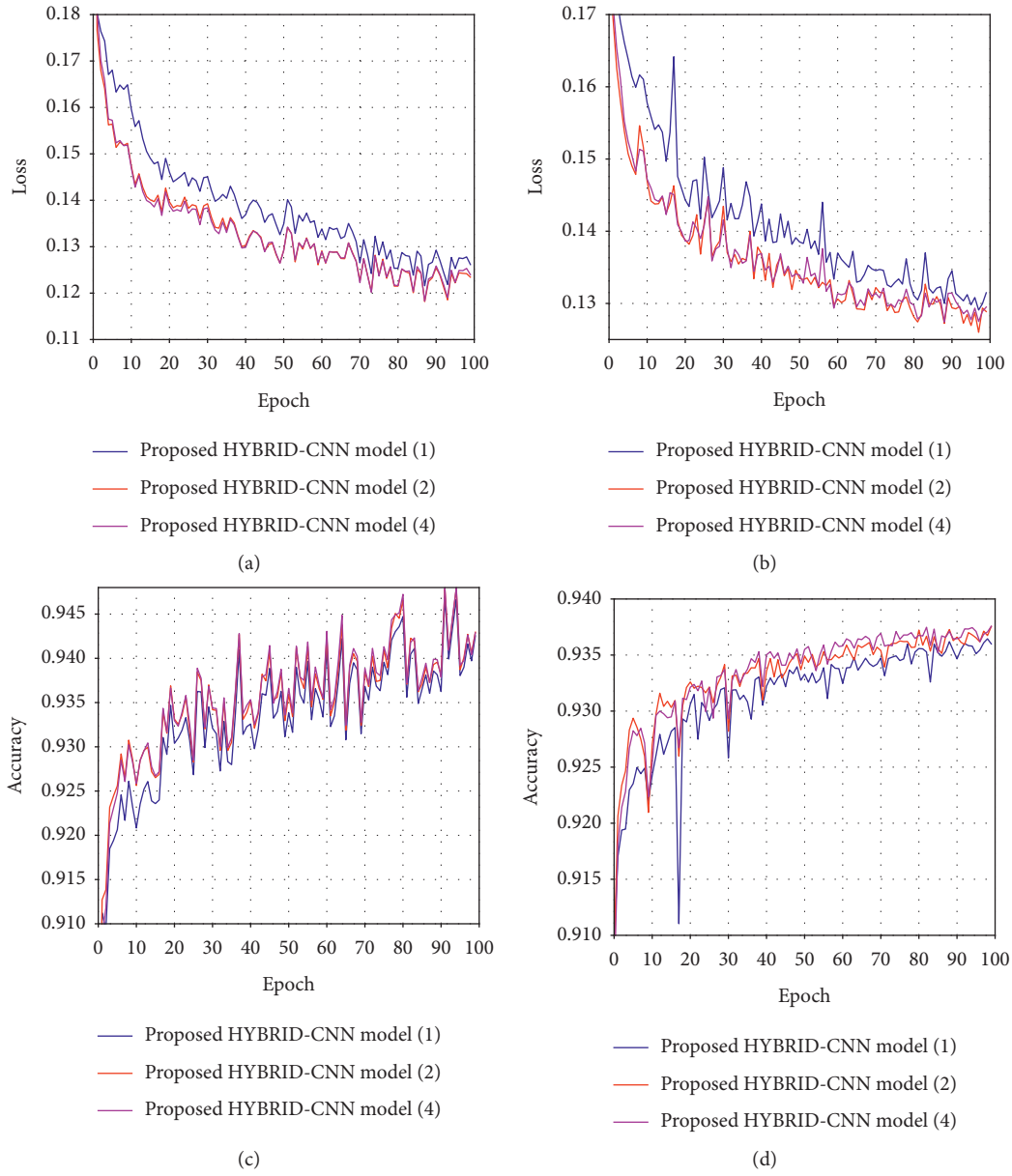

Figure 12: Parameter study of β. (a) Training loss. (b) Validation loss. (c) Training accuracy. (d) Validation accuracy.

because the model could not extract the local features of the flow, and CNN has a great impact on the results.

6.5. Attack Detection. In order to detect the attack type of abnormal flow, the dataset we used to evaluate the model was KDDCup 99 [38]. The entire dataset has approximately 5 million flow records, each of which has 41 features (the 1–9 features are the basic attributes of the packet, the 10–22 features are the packet content, and the 23–31 features are flow function and 32–41 are host-based features). As shown in Table 7, these attack flow instances can be further divided into DoS, U2R, R2L, and Probe. For the KDDCup 99 dataset, the flow sample has 41 features and a label. We cannot directly reshape a one-dimensional flow dataset into a two-dimensional matrix, so a zero feature is used here to add a dummy feature. It does not affect the result and is just for data reshaping.

We made comparisons with the current latest technology, and Figure 15 illustrates the relative comparison of our proposed abnormal flow detection algorithm with the current latest technology model. It is obvious from the obtained results that the proposed model performs better on the KDDCup 99 dataset than the existing scheme in terms of Accuracy, Detection Rate, and $F1$ score. Figure 15(a) shows the Precision evaluation of the proposed method corresponding to Normal, PROBE, DoS, U2R, and R2L data examples (99.92%, 98.11%, 99.98%, 93.81%, and 93.16%, respectively). Figure 15(b) shows the

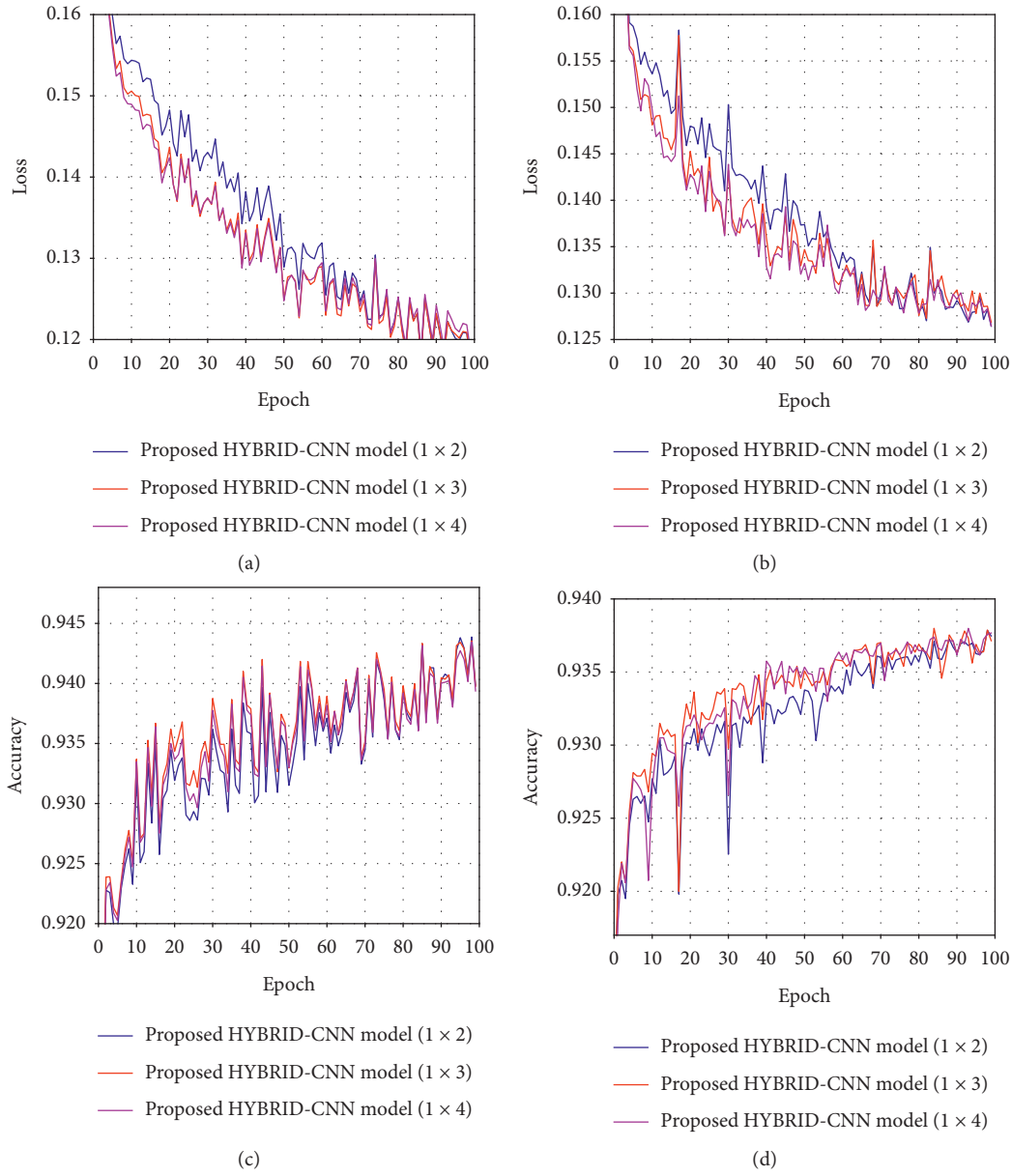

Figure 13: Parameter study of γ. (a) Training loss. (b) Validation loss. (c) Training accuracy. (d) Validation accuracy.

Detection Rate evaluation of the proposed method corresponding to successful detection of Normal, PROBE, DoS, U2R, and R2L data examples (98.21%, 93.62%, 98.89%, 92.59%, and 87.76%, respectively). Figure 15(c) shows the $F1$ score evaluation of the proposed method corresponding to Normal, PROBE, DoS, U2R, and R2L data examples (96.74%, 94.02%, 98.51%, 91.92%, and 89.37%, respectively).

It can be clearly seen from the obtained results that, for normal flow, DoS attacks and PROBE attacks have reached the maximum detection level, while detection effects for U2R and R2L attacks are slightly lower. In the real network, normal activity flow dominates while U2R and R2L are very few classes. Dataset imbalance is a quite common problem in intrusion detection. The detection model is biased towards most classes and neglects a few classes. For U2R and R2L, although the detection rate of the proposed model is lower than that of other classes, overall, it still achieves better results compared with other methods.

7. Discussion

Evaluation of the UNSW_NB15 dataset shows that our model can provide 95.64% accuracy, which is a major improvement over other deep learning methods. However, it should be noted that the results of the "R2L" and "U2L" attack classes are lower than those of other classes, because the model needs more data to learn. Unfortunately, due to the severe imbalance in the training data of such attacks, the results obtained are not stable. Hybrid detection methods are mainly combined with deep

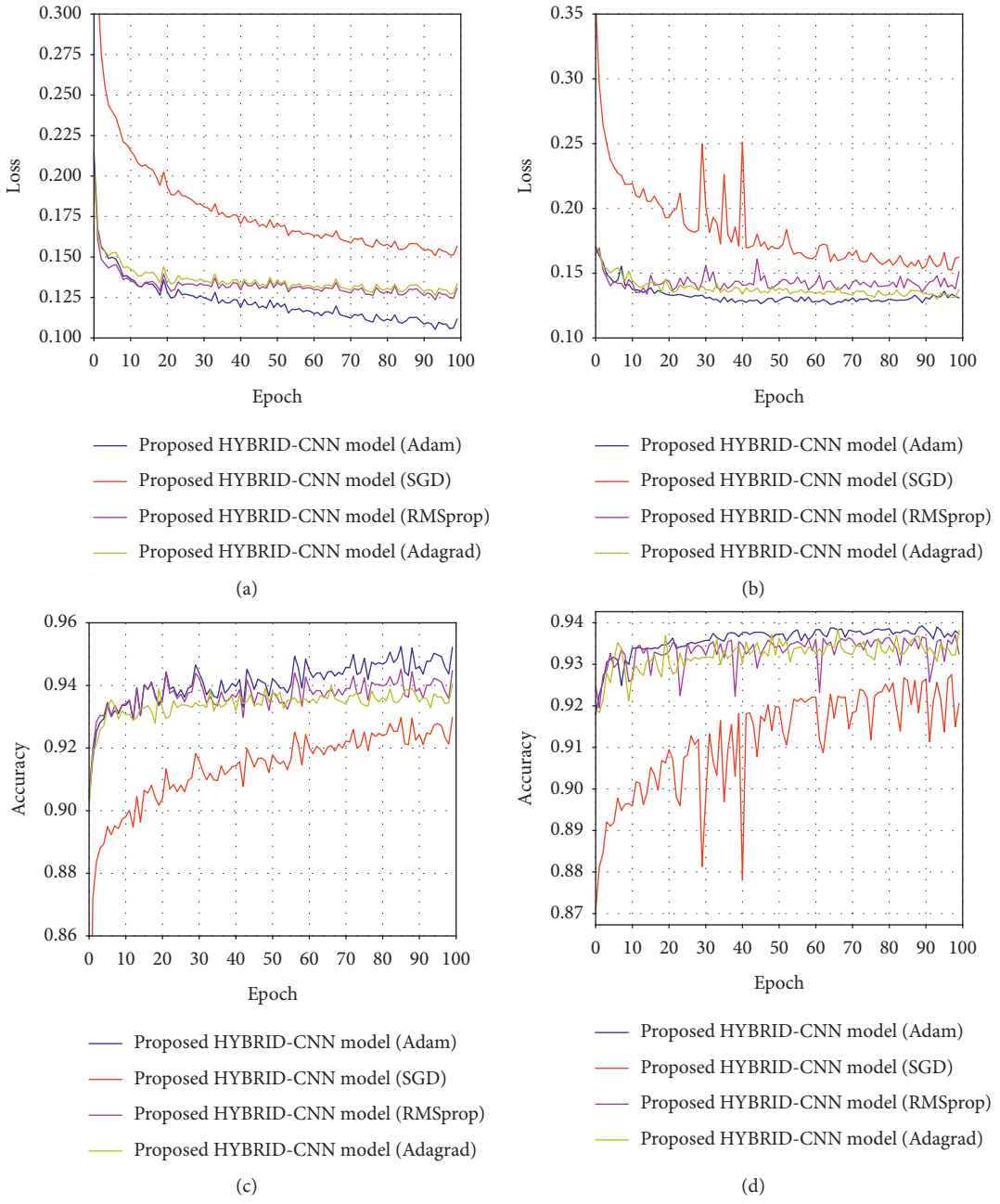

Figure 14: Parameter study of ϵ. (a) Training loss. (b) Validation loss. (c) Training accuracy. (d) Validation accuracy.

Table 6: Detailed performance (%) of HYBRID-CNN in ablation study.

Model	Acc	DR	Pre	FPR
HYBRID-CNN	**95.64**	**98.56**	**96.13**	**4.42**
(1) w/o attention	94.88	98.29	95.77	4.69
(2) w/o DNN	93.57	93.94	93.16	5.89
(3) w/o CNN	91.85	92.47	92.43	7.36

learning models, which can usually achieve higher detection accuracy. Considering the complexity of the deep learning algorithm, the algorithm can use less running time. Of course, our proposed model will spend more time on training, but using GPU acceleration can reduce training time.

Table 7: Attacks in the KDDCup 99 dataset.

Category	Training dataset	Testing dataset
DoS	Back, land, Neptune, pod, smurf, teardrop	Back, land, Neptune, pod, smurf, teardrop, mailbomb, processtable, udpstorm, apache2, worm
U2R	Buffer-overflow, loadmodule, perl, rootkit	Buffer-overflow, loadmodule, perl, rootkit, sqlattack, xterm, ps
R2L	fpt-write, guess-passwd, imap, multihop, phf, spy, warezclient, warezmaster	fpt-write, guess-passwd, imap, multihop, phf, spy, warezmaster, xlock, xsnoop, snmpguess, snmpgetattack, httptunnel, sendmail, named
Probe	ipsweep, nmap, portsweep, Satan	ipsweep, nmap, portsweep, Satan, mscan, saint

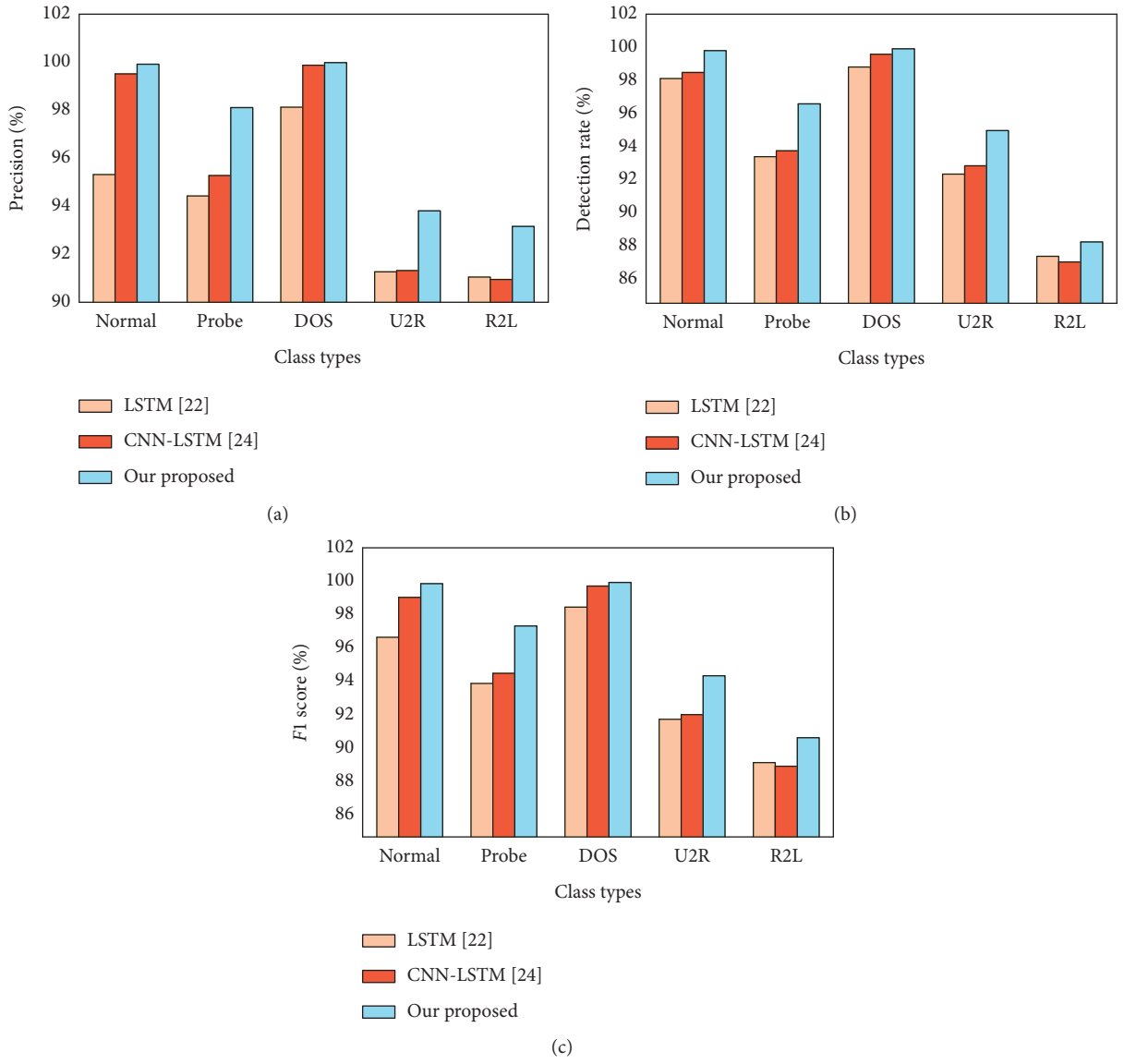

Figure 15: Experimental evaluation of the proposed method on the KDDCup 99 dataset. (a) Precision evaluation. (b) Detection Rate evaluation. (c) $F1$ score evaluation.

8. Conclusion

In this paper, we consider the problem of abnormal network flow detection of the Smart Grid integrated with the SDN. For the pursuit of accurate detection and guaranteeing network performance, we formulate a deep learning detection algorithm based on the HYBRID-CNN. In particular, our HYBRID-CNN model consists of the double channel feature extraction, key feature fusion, and classification. It gains the benefits of global memorization and local generalization brought by the DNN and the CNN, respectively. Besides, to measure the performance of the proposed algorithm, we analyze the hyperparameters of the HYBRID-CNN. Compared with other existing detection

algorithms, the experiment results show that the HYBRID-CNN has a higher detection accuracy and a lower false alarm rate.

In our future work, a problem to be solved is to improve the performance of the model through network structure optimization and automatic hyperparameter tuning. The swarm intelligent optimization algorithm, such as Particle Swarm Optimization (PSO) algorithm and Artificial Bee Colony (ABC) algorithm, can be used to automatically tune hyperparameters, which is an efficient method to improve the detection accuracy. Another problem to be solved is the unbalanced dataset. The detection accuracy of a few types of attacks needs to be improved. We hope to use data augmentation in future work to reduce the impact of the dataset.

Abbreviations

ABC: Artificial Bee Colony
CNN: Convolutional Neural Network
DNN: Deep Neural Network
FPR: False Positive Rate
IoT: Internet of Things
LSTM: Long Short-Term Memory
MLP: Multilayer Perceptron
PMU: Power Management Unit
PSO: Particle Swarm Optimization
ReLU: Rectified Linear Unit
RNN: Recurrent Neural Network
ROC: Receiver Operating Characteristic
SAE: Stacked Autoencoder
SDN: Software-Defined Network
SVM: Support Vector Machine.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61702321, 61872230, 61802249, 61802248, and U1936213).

References

[1] M. L. Tuballa and M. L. Abundo, "A review of the development of smart grid technologies," *Renewable and Sustainable Energy Reviews*, vol. 59, pp. 710–725, 2016.

[2] X. Yao, Y. Zou, Z. Chen, M. Zhao, and Q. Liu, "Topic-based rank search with verifiable social data outsourcing," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 1–12, 2019.

[3] Y. Zou, X. Yao, Z. Chen, and M. Zhao, "Verifiable keyword-based semantic similarity search on social data outsourcing," *IEEE Access*, vol. 7, pp. 5616–5625, 2018.

[4] I. Colak, S. Sagiroglu, G. Fulli, M. Yesilbudak, and C.-F. Covrig, "A survey on the critical issues in smart grid technologies," *Renewable and Sustainable Energy Reviews*, vol. 54, pp. 396–405, 2016.

[5] A. Feghali, R. Kilany, and M. Chamoun, "SDN security problems and solutions analysis," in *Proceedings of the 2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*, Paris, France, October 2015.

[6] R. Chaudhary, G. S. Aujla, S. Garg, N. Kumar, and J. J. P. C. Rodrigues, "SDN-enabled multi-attribute-based secure communication for smart grid in IoT environment," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2629–2640, 2018.

[7] M. C. Dacier, H. König, R. Cwalinski, F. Kargl, and S. Dietrich, "Security challenges and opportunities of software-defined networking," *IEEE Security & Privacy*, vol. 15, no. 2, pp. 96–100, 2017.

[8] R. L. Sahita, "State-transition based network intrusion detection," US20050111460A1, 2016.

[9] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[10] V. Vaidya, "Dynamic signature inspection-based network intrusion detection," US6279113B1, 2001.

[11] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 493–501, 2019.

[12] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, no. S1, pp. 949–961, 2017.

[13] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: a novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[14] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: mobile encrypted traffic classification using multimodal deep learning," *Computer Networks*, vol. 165, Article ID 106944, 2019.

[15] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[16] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," *Journal of Computational Science*, vol. 25, pp. 152–160, 2018.

[17] N. Ashraf, W. Ahmad, and R. Ashraf, "A comparative study of data mining algorithms for high detection rate in intrusion detection system," *Annals of Emerging Technologies in Computing (AETiC)*, vol. 2, no. 1, 2018.

[18] K. Rai, M. S. Devi, and A. Guleria, "Decision tree based algorithm for intrusion detection," *International Journal of Advanced Networking and Applications*, vol. 7, no. 4, p. 2828, 2016.

[19] R. R. Reddy, Y. Ramadevi, and K. N. Sunitha, "Effective discriminant function for intrusion detection using SVM," in *Proceedings of the 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Jaipur, India, September 2016.

[20] 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 *Proceedings of the 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, Fez, Morocco, October 2016.

[21] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: a search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[22] Y. Xin, L. Kong, Z. Liu et al., "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.

[23] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, Beijing, China, July 2017.

[24] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18042–18050, 2017.

[25] W. Wang, Y. Sheng, J. Wang et al., "Hast-ids: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2017.

[26] S. Demirci and S. Sagiroglu, "Software-defined networking for improving security in smart grid systems," in *Proceedings of the 2018 7th International Conference on Renewable Energy Research and Applications (ICRERA)*, Paris, France, 2018.

[27] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[28] C. Gonzalez, S. M. Charfadine, O. Flauzac, and F. Nolot, "SDN-based security framework for the iot in distributed grid," in *Proceedings of the 2016 International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*, Split, Croatia, July 2016.

[29] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," 2018, https://arxiv.org/abs/1803.08375.

[30] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/pdf/1412.6980.pdf.

[31] E. Bisong, "Introduction to scikit-learn," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pp. 215–229, Springer, Berlin, Germany, 2019.

[32] S. Sharma, A. Soni, and V. Malviya, "Face recognition based on convolution neural network (CNN) applications in image processing: a survey," in *Proceedings of the Recent Advances in Interdisciplinary Trends in Engineering & Applications (RAITEA)*, 2019.

[33] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10, 1995.

[34] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," in *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)*, pp. 5998–6008, Long Beach, CA, USA, 2017.

[35] K. Adem, S. Kiliçarslan, and O. Cömert, "Classification and diagnosis of cervical cancer with stacked autoencoder and softmax classification," *Expert Systems with Applications*, vol. 115, pp. 557–564, 2019.

[36] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia, November 2015.

[37] A. Tharwat, "Classification assessment methods," *Applied Computing and Informatics*, 2018.

[38] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDDCup 99 data set," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, Canada, July 2009.