Hindawi
Security and Communication Networks
Volume 2023, Article ID 9835476, 1 page
https://doi.org/10.1155/2023/9835476

Retraction

Retracted: CLD-Net: A Network Combining CNN and LSTM for Internet Encrypted Traffic Classification

Security and Communication Networks

Received 26 December 2023; Accepted 26 December 2023; Published 29 December 2023

This article has been retracted by Hindawi, as publisher, following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of systematic manipulation of the publication and peer-review process. We cannot, therefore, vouch for the reliability or integrity of this article.

Please note that this notice is intended solely to alert readers that the peer-review process of this article has been compromised.

Wiley and Hindawi regret that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] X. Hu, C. Gu, and F. Wei, "CLD-Net: A Network Combining CNN and LSTM for Internet Encrypted Traffic Classification," *Security and Communication Networks*, vol. 2021, Article ID 5518460, 15 pages, 2021.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 5518460, 15 pages
https://doi.org/10.1155/2021/5518460

Research Article

CLD-Net: A Network Combining CNN and LSTM for Internet Encrypted Traffic Classification

Xinyi Hu,[1,2] **Chunxiang Gu**,[1,2] **and Fushan Wei**[1,2]

[1]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China*
[2]*Henan Key Laboratory of Network Cryptography Technology, Zhengzhou, China*

Correspondence should be addressed to Xinyi Hu; huxinyi.1994@foxmail.com

Received 9 March 2021; Revised 3 April 2021; Accepted 10 May 2021; Published 18 June 2021

Academic Editor: Chi-Hua Chen

The development of the Internet has led to the complexity of network encrypted traffic. Identifying the specific classes of network encryption traffic is an important part of maintaining information security. The traditional traffic classification based on machine learning largely requires expert experience. As an end-to-end model, deep neural networks can minimize human intervention. This paper proposes the CLD-Net model, which can effectively distinguish network encrypted traffic. By segmenting and recombining the packet payload of the raw flow, it can automatically extract the features related to the packet payload, and by changing the expression of the packet interval, it integrates the packet interval information into the model. We use the ability of Convolutional Neural Network (CNN) to distinguish image classes, learn and classify the grayscale images that the raw flow has been preprocessed into, and then use the effectiveness of Long Short-Term Memory (LSTM) network on time series data to further enhance the model's ability to classify. Finally, through feature reduction, the high-dimensional features learned by the neural network are converted into 8 dimensions to distinguish 8 different classes of network encrypted traffic. In order to verify the effectiveness of the CLD-Net model, we use the ISCX public dataset to conduct experiments. The results show that our proposed model can distinguish whether the unknown network traffic uses Virtual Private Network (VPN) with an accuracy of 98% and can accurately identify the specific traffic (chats, audio, or file) of Facebook and Skype applications with an accuracy of 92.89%.

1. Introduction

With the rapid development of the Internet, network applications and protocols emerge in an endless stream, making the types of network traffic more complex and diverse [1], which poses certain obstacles to network traffic management. Network traffic classification and recognition is an important foundation for network detection and management and one of the key technologies for maintaining cyberspace security. The higher the recognition accuracy and the finer the classification granularity, the more reliable the technical support can be provided for network managers. Mature network traffic classification and identification technology can not only optimize network configuration and reduce network security risks but also provide better service quality according to user behavior [2]. With the frequent occurrence of data leakage, network penetration, identity theft, and ransomware incidents, countries continue to promulgate new regulations and specifications on network security, and users urgently need security and privacy, and the overall network traffic has shown a trend toward encryption. The use of traffic encryption is a double-edged sword. Although it improves and maintains the security and privacy of users, it also makes the third party in the network link unable to use Deep Packet Inspection (DPI) technology to match and screen the key fields in the traffic load [3], which has caused some obstacles to the traffic review of the firewall.

Subsequently, the rise of machine learning quickly achieved fruitful results in many areas, which aroused widespread attention from researchers. At the same time, it also inspires security personnel to introduce machine learning technology into the field of traffic analysis and analyze and study encrypted traffic from a statistical

perspective [4]. Although great results have been achieved, traditional machine learning classifiers, such as support vector machine (SVM), random forest (RF), and so on, require researchers to manually construct feature sets based on expert knowledge [5], that is, feature engineering. The construction of the feature set greatly affects the results of the classification task. In many practical situations, researchers still have insufficient experience of the problem, and key statistical features are often missed, resulting in poor model effects. In order to solve this technical bottleneck, deep learning has been developed rapidly. The deep learning technology represented by artificial neural networks can automatically mine features in feature engineering, greatly reducing human intervention and discovering associations between data that have not been discovered by humans, which has obvious advantages over machine learning.

The application of artificial neural networks in encrypted traffic classification tasks further improves the classification effect, for example, Kim et al. [6] proposed the Convolutional Neural Network and Long Short-Term Memory (C-LSTM, also called C-LSTM) model. Based on the idea of allowing neural networks to learn features completely autonomously, people's usual processing flow for network streams is to first treat the data packets in the network stream as a one-dimensional grayscale image and pass it into the CNN model for learning and transform it into a vector form. Then, use the LSTM model to perform sequence learning on the network flow and finally pass it into the fully connected network to obtain the classification result. However, this processing technology still has shortcomings and needs further research. On the one hand, the model discards some obvious features when transforming the network stream into data that can be recognized by the neural network, or the cryptanalysis knowledge acquired by people after decades of hard work is difficult to integrate with the neural network model. This will lead to a decrease in the accuracy of the classification task. In order not to decrease the accuracy rate, it may require a larger data set, or a more complex neural network, or more computing resources, or a longer time to train the model. And these cost increases may be unbearable.

In order to make up for the above shortcomings, this paper proposes an end-to-end CLD-Net model to complete the application classification task of network encrypted traffic. First, according to the research results of block cryptanalysis theory [7], we cut and recombine the payload of the network packet. When network protocols transmit data, block ciphers are commonly used for encryption. The encrypted data are a kind of pseudo-random data, which are slightly different from real random data. The research of cryptanalysis theory found that Electronic Codebook (ECB) mode block cipher has nonuniform pseudo-random defects in multiple rounds of permutation, and different groups will be encrypted according to the same encryption method. Cutting and recombining the payload of network data packets may expose this statistical law more clearly, thereby helping machine learning or deep learning to accelerate the learning of statistical features. Second, when the network flow is passed into the Long Short-Term Memory (LSTM) model for sequence learning, the data packets are treated as equally spaced elements. However, it is obvious that the time stamp interval is different during the transmission of chat information, audio, or video streams. This information is critical in traditional network stream analysis. But in models such as C-LSTM, this feature is ignored because it is difficult to integrate with neural network models. Therefore, we performed statistical analysis on packet timestamps and designed a mode to insert virtual data packets to represent this time interval, so that the CLD-Net model can perceive this interval during the training process.

The current CLD-Net model can accurately identify whether the unknown network flow uses Virtual Private Network (VPN), which application software (Facebook or Skype) is used, and which class of traffic (chats, audio, or file) it belongs to. The accuracy can reach 93%, and the VPN/non-VPN identification accuracy can reach 98%.

The research object of this paper is Internet encrypted unicast bidirectional flows, regardless of the mobile traffic. According to the specific application types of different social software in the use process, a deep learning model is used to classify a single task. The contribution of this paper is summarized as follows:

(i) An end-to-end CLD-Net model is proposed to classify network encrypted traffic. It can not only automatically extract features from the packet payload of the original stream, such as C-LSTM [6], but also add the information of the packet interval and fuse it into new features and then automatically extract features from it without feature engineering.

(ii) We use the data recombination mechanism to improve the efficiency of feature learning. By changing the number of channels to make the model more suitable for the reorganized data structure, the generated features can contain more identification information, thereby improving the classification performance.

(iii) Compared with other baseline methods, CLD-Net has higher accuracy when classifying open network traffic data.

The rest of the paper is organized as follows. In Section 2, we discuss the related work on network traffic classification. Section 3 describes the specific details of the proposed CLD-Net model. Section 4 shows the experiments and analysis of the results. Finally, the paper is concluded in Section 5.

2. Related Work

In recent years, the research on encrypted traffic has mainly used feature engineering [8–13] to find out the characteristics that best reflect the features of different classes of encrypted traffic and then classify them by selecting an appropriate classifier. Currently, the commonly used classification models are mainly divided into three types: Markov models [14–16], traditional machine learning algorithms [17–21], and deep neural network methods [6, 22–31].

2.1. Feature Engineering. Feature engineering mainly includes strategies such as feature selection and feature extraction and is a very important part of the traffic classification process. As the input of the classifier, features largely determine the quality of the classification results. In 2016, Huang et al. [10] proposed the Min-Max Ensemble Feature Selection algorithm for the 29 feature sets proposed by Moore et al. [4], which can effectively solve the problem of class imbalance and achieve higher performance in traffic classification. In 2018, Shi et al. [12] proposed a new feature optimization method EFOA based on deep learning and feature selection technology, which can provide the best and robust features for traffic classification. In 2019, Casino et al. [8] proposed HEDGE, a method to distinguish encrypted traffic from compressed traffic, using three features based on randomness tests as thresholds for distinguishing high-entropy files.

2.2. Markov Model. The Markov model can make good use of sequence features to classify traffic. In 2017, Shen et al. [15] proposed an encrypted traffic classification method based on the perceivable attributes of a second-order Markov chain. Compared with previous methods based on Markov, this method can improve the classification accuracy by about 29%. In 2020, Yao et al. [16] combined the Gaussian Mixture and Hidden Markov Model for the first time and proposed an encrypted traffic classification model MGHMM based on MOG and HMM. In mainstream traffic protocols, the classification performance is better than the latest vector quantization-based traffic classification algorithms.

2.3. Machine Learning Model. Traditional machine learning algorithms use statistical features as input and can effectively distinguish different classes of encrypted traffic through training. In 2017, Anderson et al. [17] combined the standard 22 feature sets [5] and features such as packet length and Transport Layer Security (TLS) handshake metadata to compare six commonly used algorithms: linear regression, l_1/l_2 logistic regression, decision tree, random forest classification results of integration, support vector machine, and multilayer perceptron. In 2018, Mahdavi et al. [21] proposed a new semi-supervised learning method to detect and classify the traffic of encrypted applications such as Secure Shell (SSH). This method is based on graph theory and minimum spanning tree for clustering and label propagation. Then choose the C4.5 decision tree algorithm to build the classification model. The results show that the method has accurate and suitable performance when classifying different network flows. In 2018, Liu et al. [20] applied Fast Fourier Transform (FFT) to data packet length sequence and proposed a fingerprint recognition technology based on Length-aware FFT (LaFFT) features. Using random forest classifiers in actual datasets to identify different encryption applications can reach 96.8% TPR.

2.4. Deep Learning Model. As an end-to-end model, the deep neural network method can obtain better classification results through overall optimization without excessive manual intervention. In 2017, Zhou et al. [31] proposed a traffic classification algorithm based on an improved Convolutional Neural Network (CNN), which maps the traffic data to grayscale images. As the input data of the improved CNN, compared with the traditional classification method, this method can improve the classification accuracy and reduce the classification time. In 2017, Wang et al. [32] processed traffic data into images, which were directly used as the input data of the classifier and used CNN to classify malware traffic. Experiments show that one-dimensional CNN (1D-CNN) can provide better traffic classification performance than 2D-CNN. In 2019, Zeng et al. [33] proposed a lightweight framework Deep-Full-Range based on deep learning for encrypted traffic classification and intrusion detection. This method is compared with other methods on two public datasets. The average F_1-score of encrypted traffic classification reaches 13.49%, and the average F_1-score of intrusion detection reaches 12.15%. In 2020, Yang et al. [29] proposed an end-to-end encryption application classification framework based on 1D-CNN called E2E-EACF. It only needs the payload of encrypted traffic and the time of arrival to classify encrypted flows on the public dataset WRCCDC. This method can achieve 91.00% accuracy and is better than decision trees and support vector machines.

Since there is no neural network structure specifically designed for traffic data, many researchers consider combining different neural network structures to achieve improved classification results. In 2018, Yang et al. [30] proposed two feature extraction methods based on packet length and packet arrival time and then used Autoencoder and 2D-CNN to perform classification. The results show that the performance of CNN is better than traditional machine learning baseline algorithms. In 2018, Kim et al. [6] proposed a traffic data modeling method based on CNN and LSTM network, which automatically extracts time and space information as robust features from raw data. The method can detect anomalies from web traffic. In 2019, Liu et al. [23] applied Recurrent Neural Networks (RNNs) to the problem of encrypted traffic classification and proposed a flow sequence network FS-Net based on Stacked Autoencoder (SAE). Comprehensive experiments on real datasets covering 18 applications show that FS-Net has a true positive rate (TPR) of 99.14%. In 2020, Ren et al. [28] proposed a Tree structure Recurrent Neural Network (Tree-RNN) for network encryption traffic classification model. The network traffic starts from the root node of the tree structure and is classified by multiple classifiers, which can complement each other in classification performance.

In addition to models that deal with a single classification task, multitask research is also emerging endlessly. In 2018, Huang et al. [34] first applied a deep learning multitask learning system in traffic classification and proposed a new multitask learning system based on CNN. This method can solve the tasks of malware detection, VPN packet identification, and Trojan horse classification, and the effectiveness of this learning system is verified on the public malware

dataset CTU-13 and VPN traffic dataset ISCX. In 2021, Aceto et al. [35] proposed a multimode and multitask deep learning method DISTILLER for traffic classification, which can use the heterogeneity of traffic data to overcome the performance limitations of existing traffic classification proposals based on single-mode deep learning. Based on the public dataset of encrypted traffic, DISTILLER is significantly improved compared to the single-task baseline method.

With the popularization and development of mobile networks, the classification of mobile traffic has also attracted the attention of many researchers [26, 35–40]. In 2019, Aceto et al. [36] first proposed a deep learning-based mobile traffic classifier, which can deal with encrypted traffic and reflect its complex traffic patterns. In the same year, Aceto et al. [35] also proposed a multimode deep learning framework MIMETIC for encrypted traffic classification, which can utilize the heterogeneity of traffic data to overcome the existing single-mode deep learning-based traffic classification performance limitation, and verified it on the artificially generated mobile encrypted traffic dataset. In 2020, Sang et al. [26] proposed an incremental learning framework IncreAIBMF for encrypted mobile application recognition tasks. Feature learning is performed from three different aspects, and the 192-dimension feature vector is obtained as the input of the Softmax regression classifier. Finally, the class predicted by the model is obtained. In 2020, Wang et al. [39, 40] proposed a hybrid neural network model App-Net for mobile applications (APP) traffic identification, which can learn effective features from the original TLS flows. It achieves very good performance on real datasets covering 80 applications. In 2021, Akbari et al. [38] designed a feature engineering method for generalizing encrypted web protocols and developed a neural network architecture based on stacked LSTM layers and CNN to classify mobile network traffic. It can achieve 95% service classification accuracy on real datasets, and the error rate of misclassification is reduced by nearly 50% compared with other works.

3. Model Description

Due to the particularity of traffic data, it is necessary to not only consider a series of statistical features including packet payload when classifying but also pay attention to the context relationship between packet and packet and flow and flow. At present, most of the methods with better classification results use the combination of CNN and RNN, which not only uses the ability of CNN to process high-dimensional data and learns features through convolutional layers but also uses RNN that is very effective for sequence features. It can mine the timing information and semantic information in the data, and the combination of the two can handle the flow data well. Considering that the results of 1D-CNN in [32] are better than those of 2D-CNN and that traffic data as a long sequence have better performance in the training process, LSTM, as a special RNN, can solve the gradient disappearance in the long sequence training process and gradient explosion. Based on [6], this paper proposes a model combining 1D-CNN and LSTM. Aiming at the particularity of traffic data, a feature extraction method including statistics and sequence features is proposed, and parameter selection is optimized to further improve classification performance.

This section proposes a specific classification model, which mainly includes the data preprocessing process and the neural network classification process. After the preprocessing, we also introduced our proposed method of changing the representation of the time interval in order to better represent the characteristics of different classes of data.

3.1. Preprocessing Procedure. Because the raw network traffic is usually stored in the format of .pcap or .pcapng files, it cannot be directly input to the neural network, and the neural network model requires that the length of the input data must be uniform, but the length of the raw network traffic is not uniform. Therefore, data preprocessing is required to convert the raw network traffic into a grayscale image format, which is used as the input of the model, and then the corresponding methods are called to train and evaluate the model to achieve the purpose of traffic classification. The preprocessing procedure mainly includes traffic split, traffic clean, traffic recombination, and traffic conversion.

The overall information of the preprocessing process is shown in Algorithm 1.

Lines 2–4 represent the traffic split process. Traffic split is to randomly split several flow segments with a window of 10 from the raw network traffic dataset containing thousands of packets according to the size of the flow, that is, each flow segments contains 10 consecutive packets, and then the files are divided proportionally randomly divided into a training set and a test set.

Line 7 is the traffic clean process. Traffic clean is to read the payload from the split files and then trim it to eliminate redundancy and unify the length. Each packet intercepts the first 256 bytes and fills in 0 if it is less than 256 bytes to get the original packet sequence

$$\mathbf{p}_{256} = (b_1, b_2, \ldots, b_{8\times 256}). \tag{1}$$

Line 8 is the traffic recombination process. Traffic recombination is to recombine the cleaned data according to certain rules. 256 bytes are divided into 64 bytes, 32 bytes, 16 bytes, and 8 bytes to get the packet sequence

$$\begin{aligned} \mathbf{p}_\alpha &= \left(\mathbf{s}_1^\alpha, \mathbf{s}_2^\alpha, \ldots, \mathbf{s}_{(256/\alpha)}^\alpha\right), \\ \mathbf{s}_i^\alpha &= \left(b_{i,1}^\alpha, \ldots, b_{i,j}^\alpha, \ldots, b_{i,8\alpha}^\alpha\right), \end{aligned} \tag{2}$$

where $\mathbf{p}_\alpha$ represents a packet sequence of length 256 bytes and block size α bytes, $\mathbf{s}_i^\alpha$ represents the i-th block of packet sequence $\mathbf{p}_\alpha$, $b_{i,j}^\alpha$ is the j-th bit of packet sequence block $\mathbf{s}_i^\alpha$, the block size $\alpha \in \{8, 16, 32, 64, 256\}$, the block $i \in \{1, 2, \ldots, (256/\alpha)\}$, and the bit in the packet sequence $j \in \{1, 2, \ldots, 8\alpha\}$. Then, $\mathbf{p}_\alpha$ is represented by $b_{i,j}^\alpha$ as follows:

Input: raw network traffic dataset D; number of traffic classes C;
Output: packet matrix P;
(1) **for** each $i \in [1, C]$ **do**
(2) Randomly select N consecutive 10 packets in D;
(3) 10 consecutive packets consist a flow F, that is, the number of F is N;
(4) Randomly divide the F into a training set F^{train} and a testing set F^{test} in proportion; // Traffic split;
(5) **for** each $j \in [1, N]$ **do**
(6) **for** each packet $p \in F$ **do**
(7) Trim p and uniform length of 256 bytes, $p \longrightarrow$ payload $\mathbf{p}_{256}$; // Traffic clean;
(8) Recombine $\mathbf{p}_{256} \longrightarrow \widehat{\mathbf{p}}_\alpha$; // Traffic recombination;
(9) Generate the grayscale image format to get the packet matrix, $\widehat{\mathbf{p}}_\alpha \longrightarrow P$; // Traffic conversion;
(10) **end for**;
(11) **end for**;
(12) **end for**;
(13) **return** Packet matrix;

Algorithm 1: Preprocessing algorithm.

$$\widehat{\mathbf{p}}_\alpha = \left(b^\alpha_{1,1}, \ldots, b^\alpha_{j,1}, \ldots, b^\alpha_{(256/\alpha),1}, \ldots, b^\alpha_{1,8\alpha}, \ldots, b^\alpha_{i,8\alpha}, \ldots, b^\alpha_{(256/\alpha),8\alpha}\right). \tag{3}$$

Line 9 is the traffic conversion process. Traffic conversion is to convert packet sequence into the input format of the neural network. The 4 sequences obtained after recombination and the original sequence are all converted to the decimal system, and the value of each byte is 0–255 to obtain a 5×256-dimension matrix $P^T = (\mathbf{p}_{256}, \widehat{\mathbf{p}}_{64}, \widehat{\mathbf{p}}_{32}, \widehat{\mathbf{p}}_{16}, \widehat{\mathbf{p}}_8)$, and P is equivalent to a grayscale image. Input P into a 5-channel CNN, and each channel inputs a 1×256-dimension sequence $\mathbf{p}$.

3.2. Representation of Time Interval. The payload and time of the data packet are two basic features that distinguish traffic data, and most existing studies are based on these two (or one of them). Regarding the payload of the data packet, we have truncated, divided, and recombined it in the preprocessing process described in Section 3.1 and obtained a matrix that can represent the payload of the data packet. In terms of time, we consider to add the time interval between packets in the preprocessed data. The time interval between packets refers to the time interval between the arrival of two adjacent packets of the same class. Consider changing the representation of this time information so that it is input into the model in a format similar to the preprocessed data, and the payload information and time interval information are fused.

First, the time intervals of different classes of packets are counted, and then a different number of blank packets are inserted between the packets to represent different time intervals according to the counting results.

As we can see from Figure 1, in most different types of network flows, data packets with short time intervals account for the vast majority (within 1s), and the curve shape is close; however, in the interval of 1–5 s, there is a large distribution difference; the data packets with an interval more than 5 s are very few and can be ignored. Therefore, when we integrate the time interval information of network packets into the model, we insert a blank data packet for those with a data packet interval of more than 1 s to indicate that the flow has a time interval of more than 1 s. It should be noted that the blank data packet does not use a sequence of all 0 s to represent the payload but all 1 s. This is to prevent the parameters of each neuron in the neural network from being multiplied by 0 and becoming invalid when encountering a blank data packet. The 0–0.1 second part of the figure is not shown in detail because the number of data packets is too large, and the detailed representation will make other packets unobvious.

A matrix after preprocessing represents a packet. Therefore, the inserted blank data packet is a matrix P_0 of all 1 s with dimensions of 5×256, that is,

$$P_0 = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}_{5\times256}.$$

The specific formalization process is shown in Algorithm 2.

Line 3 is a process of counting the time interval of packets. Lines 4–7 represent the process of judging, that is, judging whether a blank packet should be added between two adjacent packets.

3.3. CLD-Net Model. The overall structure of the CLD-Net model is shown in Figure 2 (excluding input and output), which is mainly divided into three parts: CNN, LSTM, and several concatenation of Dense layers (also called fully connected layers). First, in CNN, the matrix P passes through Relu and the pooling layer and changes from 5×256 to 5×244. The window of the LSTM is 10, that is, 10 consecutive packets of the same flow are sequentially input into the LSTM after passing through the CNN. After the output, it enters DenseNet to perform feature dimensionality reduction. After Relu and Dropout, the matrix dimension is continuously reduced, and finally 8 classifications are performed through the Softmax layer.

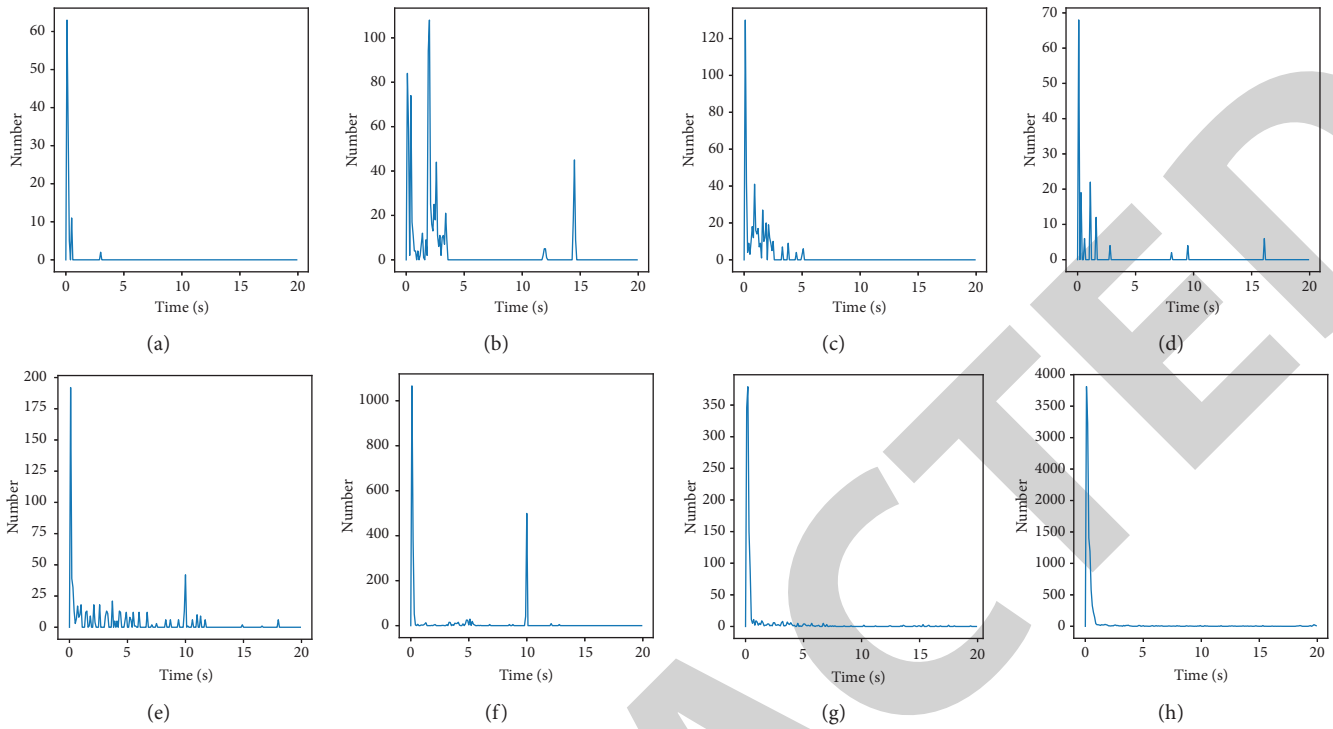

Figure 1: Time intervals for different classes of packets. (a) Facebook-chat. (b) Facebook-audio. (c) Skype-chat. (d) Skype-file. (e) VPN-Facebook-audio. (f) VPN-Facebook-chat. (g) VPN-Skype-chat. (h) VPN-Skype-files.

Input: number of classes C; packet number of one class n; packet matrix $P_1, P_2, \ldots, P_n, \ldots$;
(1) **for** each $i \in [1, C]$ **do**
(2) **for** $j = 1$ to $n - 1$ **do**
(3) Count the time interval of P_j and P_{j+1};
(4) **if** the time interval between P_j and $P_{j+1} <$ 1 s **then**
(5) Input P_{j+1} after P_j to the CLD-Net model;
(6) **else**
(7) Add P_0 between P_j and P_{j+1} and input them to the CLD-Net model in turn;
(8) **end if**
(9) **end for**;
(10) **end for**;

Algorithm 2: Formalization of the representation of time interval.

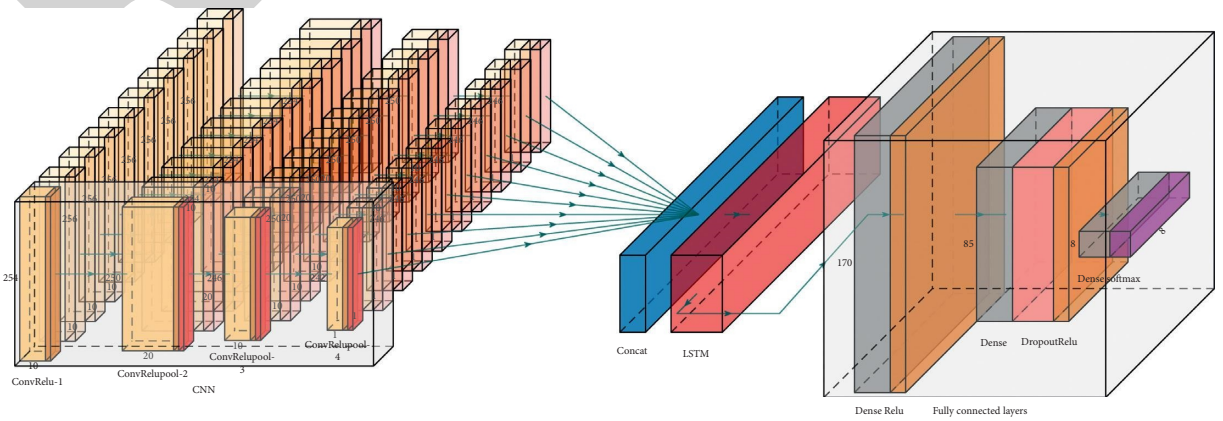

Figure 2: The basic framework of CLD-Net.

The left part of Figure 2 is a CNN, which contains 10 four-layer 1D-CNN. For each CNN, the input is a 5×256-dimension matrix, where 5 is the number of channels. Assume that the input of each channel is $\mathbf{x} = (x_1, x_2, \ldots, x_{256})$, i is the index of the feature, and j is the index of the feature map. The output of the first convolutional layer is

$$\begin{aligned} \mathrm{Out}_{ij}^{1} &= \mathrm{Relu}\left(\mathrm{Conv}_{ij}^{1}\right), \\ \mathrm{Conv}_{ij}^{1} &= \sum_{m=1}^{3} W_{m,j}^{0} x_{i+m-1,j}^{0} + B_{j}^{0}, \end{aligned} \tag{4}$$

where W is the weight, B is the bias, and 3 is the kernel size. The number of output channels is 10; then, the output of the first layer is a 10×254-dimension matrix. The output of the second convolutional layer is

$$\begin{aligned} \mathrm{Out}_{ij}^{2} &= \mathrm{Relu}\left(\mathrm{MaxPool}_{ij}^{2}\right), \\ \mathrm{MaxPool}_{ij}^{2} &= \max_{r \le 3}\left(\mathrm{Conv}_{i\times 1+r,j}^{2}\right), \end{aligned} \tag{5}$$

where 1 is the stride, 3 is the pooling size, and

$$\mathrm{Conv}_{ij}^{2} = \sum_{m=1}^{M} W_{m,j}^{1} x_{i+m-1,j}^{1} + B_{j}^{1}. \tag{6}$$

The number of output channels is 20; then, the output of the second layer is a 20×250-dimension matrix. The third and fourth layers are the same as the second layer, and the number of output channels is 10 and 1; . Finally, CNN outputs ten 1×242-dimension vectors.

The middle part of Figure 2 is the Concat layer and LSTM. The Concat layer is used to stitch together 10 consecutive CNN outputs and input them into the LSTM in turn. LSTM mainly includes input gates, forget gates, and output gates. Assuming that the input of LSTM at time t is X_t, it is known that the cell state at the previous time is C_{t-1} and the hidden state is S_{t-1}, and we get the input gate, forget gate $F(t)$, cell state C_t, and output gate $O(t)$ at time t, where the input gate includes the sigmoid function $I(t)$ and the tanh function $R(t)$:

$$\begin{aligned} I(t) &= \mathrm{sigmoid}\left(W_i^T S_{t-1} + U_i^T X_t + B_i\right), \\ R(t) &= \tanh\left(W_r^T S_{t-1} + U_r^T X_t + B_r\right), \\ F(t) &= \mathrm{sigmoid}\left(W_f^T S_{t-1} + U_f^T X_t + B_f\right), \\ C_t &= C_{t-1} * F(t) + I(t) * R(t), \\ O(t) &= \mathrm{sigmoid}\left(W_o^T S_{t-1} + U_o^T X_t + B_o\right), \end{aligned} \tag{7}$$

where W and U are the weight matrices of hidden state and input, respectively, and B is the bias matrix. The final output is

$$S_t = \tanh\left(C_t\right) * O(t). \tag{8}$$

The entire LSTM reduces 10 242-dimension vectors to 170 dimensions, which is about 0.7 of the original.

The right part of Figure 2 consists of three fully connected layers. It acts as a "classifier" in the entire model. CNN and LSTM map the raw data to the hidden layer feature space, and the fully connected layer maps the learned "distributed feature representation" to the sample label space. Set the output of the first Dense (fully connected) layer to half of the input, that is, an 85-dimension vector. Add a Dropout layer after the second Dense layer and remove the training unit from the network according to a certain probability to prevent overfitting, so as to improve the generalization ability of the model. The output is an 8-dimension vector for the final classification. Add a Softmax classifier after the third Dense layer:

$$P\left(y_i \mid x_i, w\right) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}. \tag{9}$$

Given the input x_i and the parameter w, the normalized probability assigned to the correct classification label is obtained, and the data are divided into 8 classes according to the result to complete the classification task of the model.

4. Experiment

This section mainly introduces the data used in the experiment, the metrics for evaluating and validating the performance of the experiment, and the specific experimental results and analysis.

4.1. Dataset Description. This paper uses the ISCX public traffic dataset [41]. The experimental data are composed of the traffic of Facebook and Skype, two social APPs, and classified according to the specific uses. Facebook traffic includes chat and audio transfer, Skype traffic includes chat and file transfer, and each class chooses the network traffic that uses VPN for protocol encapsulation and the ordinary network traffic NoVPN. There are a total of 8 classes of data, and the numbers of each class are shown in Table 1.

A total of 8996 data packets were used in the experiment, 8097 were selected as the training set in proportion, and the remaining 899 were used as the test set.

4.2. Evaluation and Validation Metrics. The two-class problem evaluation and validation metrics are usually formulated in four cases: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). TP is the case where a positive sample is correctly classified as a positive sample; FP is the case where a negative sample is incorrectly classified as a positive sample; TN is the case where a negative sample is correctly classified as a negative sample; and FN is the case where a positive sample is incorrectly classified as a negative sample. According to the above four cases, we use four basic metrics for evaluation and validation: accuracy, precision, recall, and F_1-score.

Table 1: Data classes and packet number of different classes.

	NoVPN	VPN
Facebook-chat	1258	3059
Facebook-audio	711	403
Skype-chat	991	324
Skype-file	572	1678
Total	3532	5464

Multiclass problem usually uses macro-accuracy for evaluation and validation, that is, the proportion of correctly classified samples to the total number of samples.

Accuracy refers to the proportion of correctly classified samples n_{correct} to the total number of samples n_{total}. It serves as a metric that can evaluate both two-class problem and multiclass problem, given by the following equation:

$$\text{accuracy} = \frac{n_{\text{correct}}}{n_{\text{total}}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{10}$$

Precision refers to the proportion of the number of positive samples that are correctly classified to the number of samples that the classifier judges to be positive, which is used to evaluate two-class problem and is given by

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{11}$$

Recall refers to the proportion of the number of correctly classified positive samples to the number of true positive samples, used to evaluate two-class problem, which is given by

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{12}$$

F_1-score refers to the harmonic mean value of precision and recall, which is used to evaluate two-class problem and is given by

$$F_1 - \text{score} = 2 \times \frac{\text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}} = \frac{2\,\text{TP}}{2\,\text{TP} + \text{FP} + \text{FN}}. \tag{13}$$

4.3. Results and Analysis. The experiment uses ten-fold cross test, each experiment is repeated ten times, and the results are averaged.

The experimental computing resource information is roughly as follows: AliCloud server ecs.gn6i instance with a 4-core vCPU processor, 15 GB memory, and an NVIDIA T4 GPU.

First, for the two-class problem (VPN or NoVPN), we counted TP, TN, FP, and FN and calculated accuracy, precision, recall, and F_1-score based on the evaluation metrics. The results are shown in Figure 3.

As can be seen from the figure, for distinguishing whether it is VPN data, the average accuracy, recall and F_1-score all exceed 0.97, and the average precision even exceeds 0.98. The results of the ten experiments are relatively stable, basically between 0.96 and 0.99, with little difference. The larger the value, the better the classification result, which means that TP and TN are significantly higher than FP and FN (see Table 2 for specific data).

For the eight-class problem, we count accuracy in Figure 4.

It can be seen from Figure 4 that the accuracies of the ten experiments are above 0.91, the highest is over 0.95, the average accuracy is close to 0.93, and the results of the ten experiments are very stable. This shows that the model we proposed has a good effect on distinguishing whether the data are VPN data and classifying specific applications and classes.

It can be seen from Figure 5 that the accuracy increased rapidly in the first 10 epochs. After 10th epoch, the accuracy tends to stabilize. After about 63rd epoch, the accuracy remains above 0.92. This shows that the CLD-Net model training process is robust and the results are good. Figures 3 and 4 show that the test results are good and the model can effectively distinguish different classes of Internet encrypted traffic.

4.3.1. Comparison with Baseline Methods. In order to verify that the deep learning-based method we proposed is better than the traditional machine learning method, we chose the support vector machine (SVM) and random forest (RF) to conduct an experimental comparison of eight-class problem. The feature extraction method of machine learning is similar to Section 3.1. The data packets are divided into blocks of 1 byte, 8 bytes, 16 bytes, 32 bytes, and 64 bytes and then recombined. Figure 6 shows the experimental results of SVM and RF, which are divided into different blocks. It is also compared with the results of CLD-Net.

It can be seen from Figure 6 that SVM and RF will have slightly different results according to different blocks. The best result in SVM is the case of 16 bytes, which is also the best result among the eight cases of SVM and RF. The best result in RF is the case of 32 bytes, which is not much different from the case of 16 bytes. In general, the larger the block, the better the result, but the average accuracy is lower than 0.9. The average accuracy of CLD-Net is above 0.92, which is significantly better than these two traditional machine learning methods.

4.3.2. Comparison with C-LSTM. C-LSTM is the method proposed in [6], and it is also a method of combining CNN and LSTM. We reproduce the model in [6] and use the same raw dataset as CLD-Net for an eight-class experimental comparison. The results are shown in Figure 7.

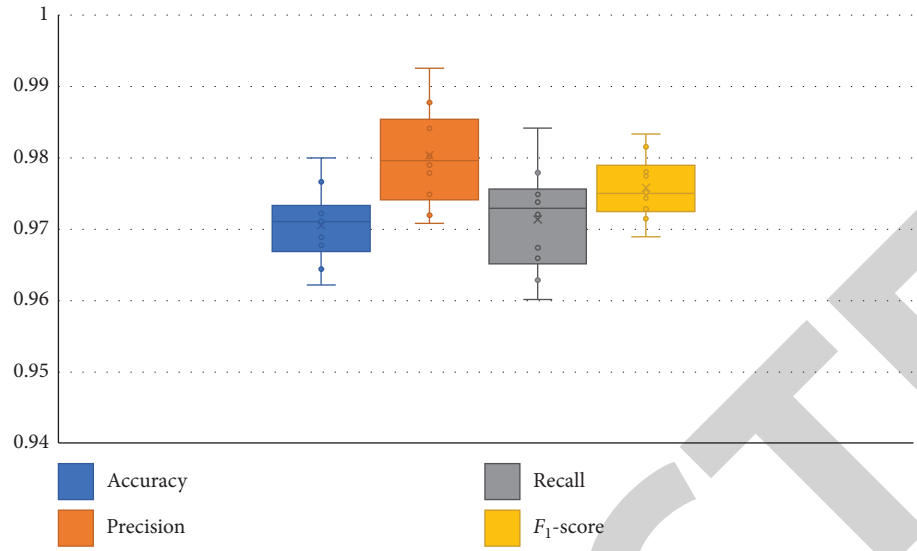

Figure 3: Experimental results of the two-class problem.

Table 2: Specific experimental data for two-class problem.

No.	TP	TN	FP	FN
1	557	317	9	16
2	504	369	13	13
3	531	350	4	14
4	545	322	11	21
5	510	363	8	18
6	530	335	12	22
7	559	319	12	9
8	564	309	7	19
9	520	350	15	14
10	532	339	16	12
Ave	535.2	337.3	10.7	15.8

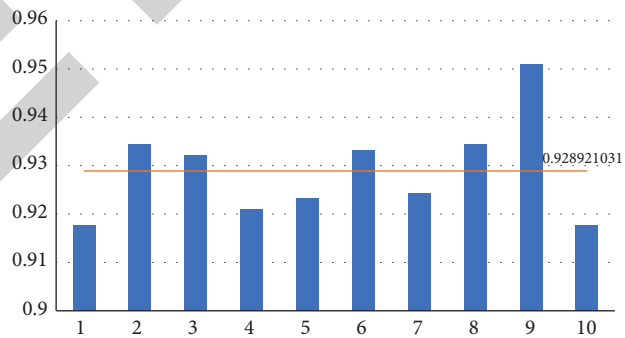

Figure 4: Accuracy of the eight-class problem.

It can be seen from Figure 7 that the results of our proposed CLD-Net are significantly better than those of C-LSTM, and the average accuracy is about 5 percentage points higher.

It can be seen from Figure 8 that both models can clearly classify 8 classes (the red color block is at the diagonal position and the color is darker). But the error rate of our proposed CLD-Net model is lower (the yellow color blocks appear less and the color is light). The C-LSTM model has a higher probability of classification errors. In other parts except the diagonal position, there are many yellow blocks, and there are darker yellow blocks adjacent to the diagonal. This means that when distinguishing different specific applications of the same APP and the

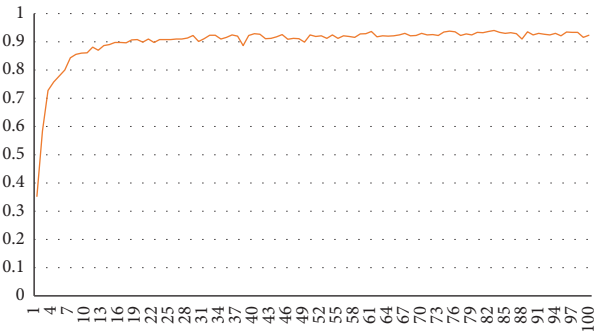

Figure 5: Accuracy of 100 epochs during training process.

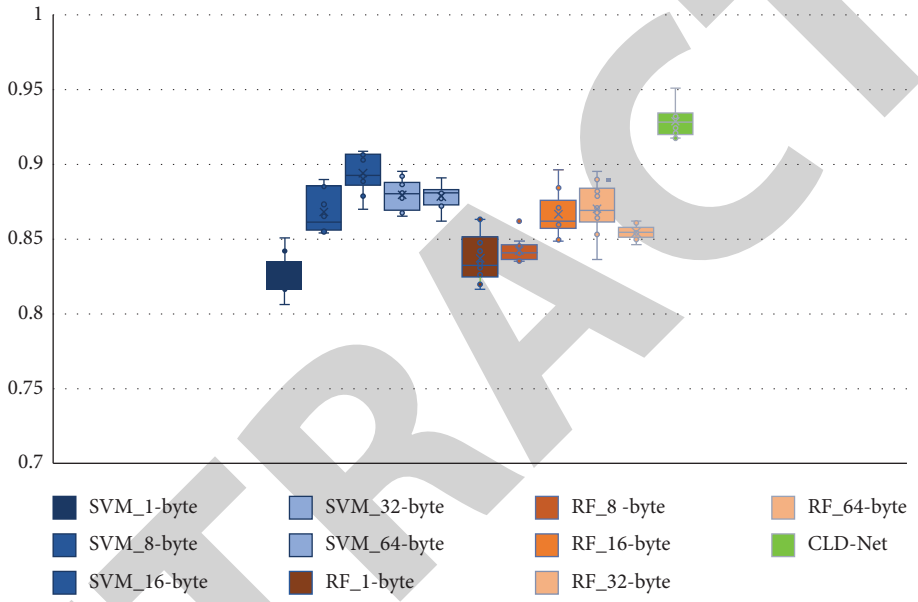

Figure 6: Comparison of baseline method and CLD-Net.

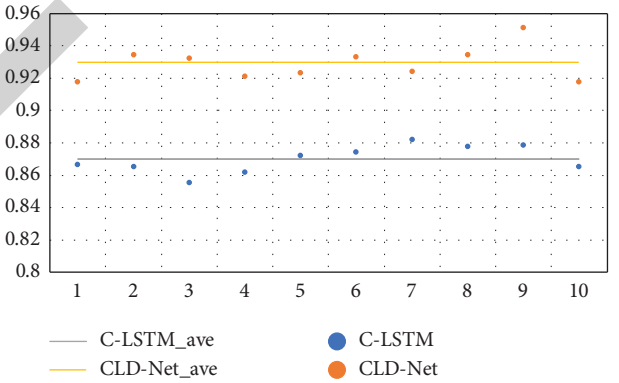

Figure 7: Comparison of CLD-Net and C-LSTM.

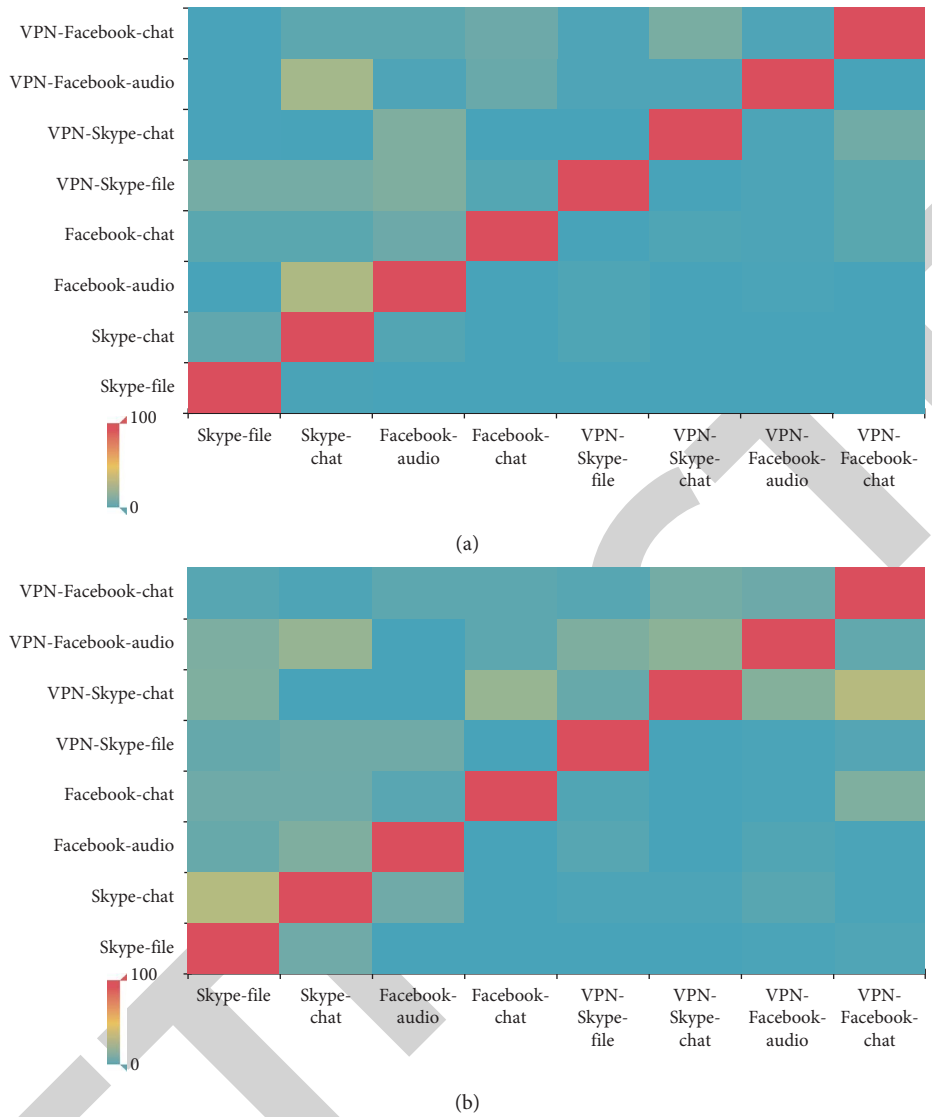

Figure 8: The confusion matrix of the eight-class problem: (a) CLD-Net; (b) C-LSTM. The color changes from cool color (blue) to warm color (red), representing the accuracy rate from 0 to 100%. The darker the color, the larger the percentage.

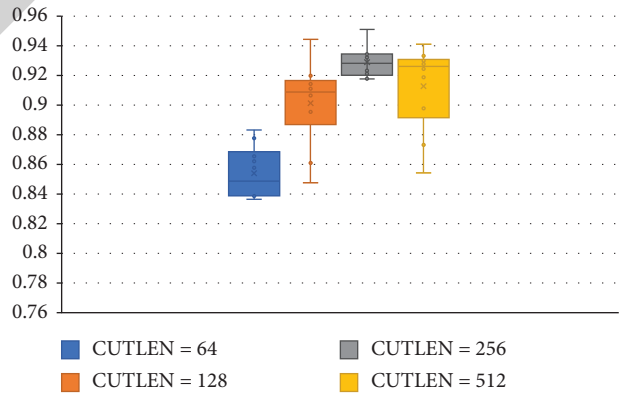

Figure 9: Comparison of results with different truncation lengths.

Table 3: Time record of 10 runs of CLD-Net eight-class problem under different truncation lengths (unit: second).

No.	CUTLEN = 64	CUTLEN = 128	CUTLEN = 256	CUTLEN = 512
1	1005.041	1067.144	1034.527	1196.198
2	1002.892	1067.637	1035.191	1193.369
3	1002.318	1066.098	1035.481	1193.317
4	1002.496	1067.72	1035.533	1195.855
5	1003.004	1066.287	1035.174	1191.928
6	1002.779	1066.163	1037.301	1192.863
7	1004.311	1068.104	1036.128	1198.398
8	1002.475	1065.489	1036.616	1191.378
9	1003.534	1071.188	1036.902	1192.536
10	1003.149	1067.685	1035.806	1195.579
Ave	1003.1999	1067.3515	1035.8659	1194.1421

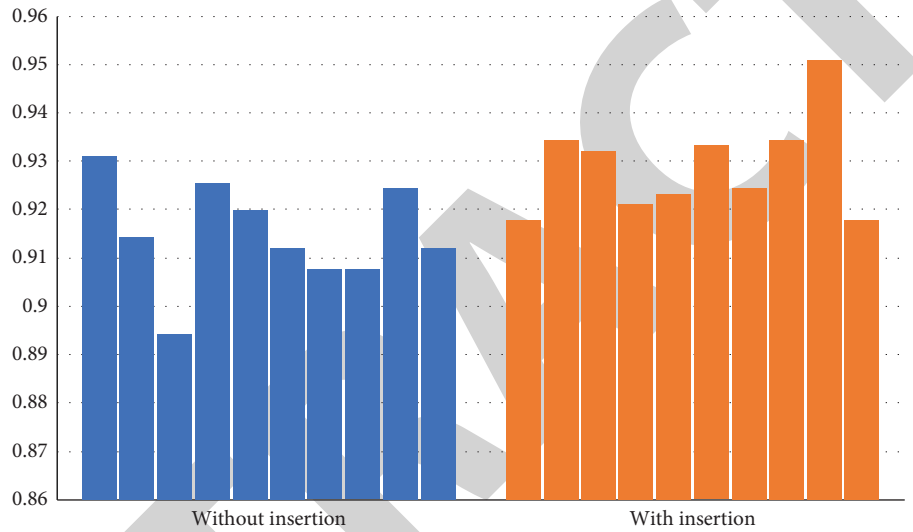

Figure 10: Comparison of the original model with not inserting blank data packets.

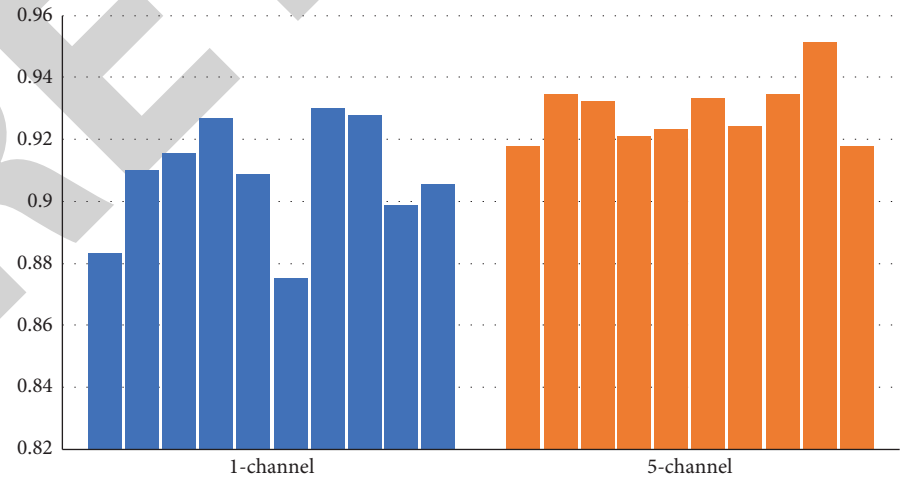

Figure 11: Comparison of results for different channel numbers.

same specific application of different APPs, the probability of error is higher.

In addition to considering the comparison of classification accuracy, the comparison of time complexity is also considered. Compared with C-LSTM, the time complexity of CLD-Net mainly changes in the following three aspects:

(1) Increase of channel number and truncation length: the time complexity increases from the original $O(n)$ to $O((c/c_0)\cdot(l/l_0)\cdot n)$, where c is the channel number, $c = 5, c_0 = 1$, l is the truncation length, $l = 256$, and $l_0 = 60$; then,

$$O\left(\frac{c}{c_0}\cdot\frac{l}{l_0}\cdot n\right) = O\left(\frac{5}{1}\cdot\frac{256}{60}\cdot n\right) \approx O(5\cdot 4\cdot n) = O(20n). \tag{14}$$

However, in practical applications, the increase of channel number and truncation length can be optimized in parallel to reduce time loss.

(2) Inserting the time interval leads to an increase in the data size: there are a total of 8996 samples, and each window is 10, which is 89,960 data packets. After inserting a blank packet, the total length becomes 94,405, and the data size increases by approximately 4.7%.

(3) Increase of the neuron number: it can be accelerated by GPU, and the impact is negligible.

It can be seen that compared with C-LSTM, the time complexity of CLD-Net will not increase much.

Compared with C-LSTM, CLD-Net has three main differences besides the specific settings of the neural network: the truncation length of data packets, the utilization of the time interval between data packets, and the data recombination in the preprocessing process. Regarding these three differences, we will verify the advantages of the differences in the internal analysis of the model in Section 4.3.3.

4.3.3. Internal Analysis of CLD-Net Model. The data packet in [6] is truncated by 60 bytes. After experiments, we found that 60 bytes is not the optimal truncation length. We tested truncated 64 bytes, 128 bytes, 256 bytes, and 512 bytes in turn. The results are shown in Figure 9.

It can be seen from Figure 9 that the result is best when the truncation length is 256 bytes, and the rest are 512 bytes, 128 bytes, and 64 bytes in order. On the whole, the longer the truncation length, the higher the classification accuracy. However, a too long truncation length will increase time and waste computing resources.

It can be seen from Table 3 that if the truncation length is increased from 64 bytes to 512 bytes, the running time will increase by nearly 2 minutes. The running time with a truncation length of 256 bytes is shorter than that of 128 bytes and 512 bytes. According to the compromise between Figure 9 and Table 3, we choose 256 bytes as the truncation length of the data packets.

In the C-LSTM model of [6], only the basic feature of packet payload is used, and the time interval between packets is not considered. If we does not add time interval information in CLD-Net model, that is, does not insert a 5×256-dimensional matrix of all 1 s, and the experimental results are shown in Figure 10.

It can be seen from Figure 10 that the average accuracy of not inserting blank data packets, that is, not adding time intervals feature, is about 0.915, which is lower than 0.929 of the original model, and the results of ten experiments are basically lower than those of the original model. Therefore, the time interval as a basic feature can effectively improve the accuracy of model classification.

The data after preprocessing as described in Section 3.1 will be input to a 5-channel CNN. If the number of channels is changed to 1, the experimental results are shown in Figure 11.

5. Conclusion

In this paper, we propose a novel Internet encrypted traffic classification model. Compared with traditional traffic classification methods, this model does not require complicated feature extraction and selection. In this model, through the flow segmentation and recombination of the preprocessing process, the effective use of packet payload information is ensured, and the expression of the time interval between packets is transformed to achieve more information utilization. Using CNN and LSTM can efficiently and automatically learn the statistical features and sequence features of traffic. According to the results of the public dataset, it is proved that the performance of CLD-Net is significantly improved compared with the latest methods.

In future work, we can consider using this traffic classification model for malware traffic identification, intrusion detection, and other network security fields. At the same time, we should also consider not only classifying offline traffic but also dealing with online real-time network encryption traffic. According to the unexpected situation of the network, we should improve and update the model in time for different traffic data distribution situations to achieve higher adaptability. In addition, considering the real-world data distribution, there may be noise interference, unbalanced datasets, and inaccurate ground truth [17]. In order to further eliminate these possible problems and eliminate the impact on classification performance, we will conduct corresponding research in the next stage.

Data Availability

This paper uses the ISCX public traffic dataset which is available in [41].

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported in part by the National Natural Science Foundation of China under grant no. 61772548.

References

[1] Paloalto, *Real-World Threat & Application Reporting*, 2015, https://www.paloaltonetworks.com/resources/research/application-usage-and-threat-report.

[2] Y. Wang, H. Zhou, H. Feng, Y. Miao, and W. Ke, "Network traffic classification method basing on CNN," *Journal of Communication*, vol. 39, no. 1, pp. 14–23, 2018.

[3] A. Dainotti, A. Pescapé, and K. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, pp. 35–40, 2012.

[4] A. Moore and D. Zuev, "Internet traffic classification using Bayesian analysis techniques," in *Proceedings of the SIGMETRICS '05*, Banff, AB, Canada, June 2005.

[5] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.

[6] T. Y. Kim and S. B. Cho, "Web traffic anomaly detection using C-LSTM neural networks," *Expert Systems with Applications*, vol. 106, pp. 66–76, 2018.

[7] X. Hu and Y. Zhao, "Block ciphers classification based on random forest," *Journal of Physics Conference Series*, vol. 1168, no. 3, Article ID 032015, 2019.

[8] F. Casino, K.-K. R. Choo, and C. Patsakis, "HEDGE: efficient traffic classification of encrypted and compressed packets," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 11, pp. 2916–2926, 2019.

[9] Y. Chen, T. Zang, Y. Zhang, Y. Zhou, and Y. Wang., "Rethinking encrypted traffic classification: a multi-attribute associated fingerprint approach," in *Proceedings of the 2019 IEEE 27th International Conference on Network Protocols (ICNP)*, pp. 1–11, Chicago, IL, USA, October 2019.

[10] Y. Huang, Y. Li, and B. Qiang, "Internet traffic classification based on min-max ensemble feature selection," in *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3485–3492, Vancouver, BC, Canada, July 2016.

[11] M. Lotfollahi, R. S. H. Zade, M. Jafari Siavoshani, and M. Saberian, "Deep packet: a novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, pp. 1999–2012, 2020.

[12] H. Shi, H. Li, D. Zhang, C. Cheng, and X. Cao, "An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification," *Computer Networks*, vol. 132, pp. 81–98, 2018.

[13] Z. Zhang, C. Kang, P. Fu, Z. Cao, Z. Li, and G. Xiong, "Metrie learning with statistical features for network traffic classification," in *Proceedings of the 2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*, pp. 1–7, San Diego, CA, USA, December 2017.

[14] L. Chang, Z. Cao, G. Xiong, G. Gou, S. Yiu, and L. He, "MaMPF: encrypted traffic classification based on multi-attribute Markov probability fingerprints," in *Proceedings of the 2018 IEEEACM 26th International Symposium on Quality of Service (IWQoS)*, pp. 1–10, Banff, AB, Canada, June 2018.

[15] M. Shen, M. Wei, L. Zhu, and M. Wang, "Classification of encrypted traffic with second-order Markov chains and application attribute bigrams" *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1830–1843, 2017.

[16] Z. Yao, J. Ge, Y. Wu, X. Lin, R. He, and Y. Ma, "Encrypted traffic classification based on Gaussian mixture models and Hidden Markov Models," *Journal of Network and Computer Applications*, vol. 166, no. 1, Article ID 102711, 2020.

[17] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Halifax, NS, Canada, August 2017.

[18] K. L. Dias, M. A. Pongelupe, W. M. Caminhas, and L. de Errico, "An innovative approach for real-time network traffic classification," *Computer Networks*, vol. 158, pp. 143–157, 2019.

[19] S. Leroux, S. Bohez, P.-J. Maenhaut, M. Nathan, P. Simoens, and B. Dhoedt, "Fingerprinting encrypted network traffic types using machine learning," in *Proceedings of the NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–5, Taipei, Taiwan, April 2018.

[20] L. Chang, Z. Cao, Z. Li, and G. Xiong, "LaFFT: length-aware FFT based fingerprinting for encrypted network traffic classification," in *Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, Natal, Brazil, June 2018.

[21] E. Mahdavi, A. Fanian, and H. Hassannejad, "Encrypted traffic classification using statistical features," *The ISC International Journal of Information Security*, vol. 10, no. 1, pp. 29–43, 2018.

[22] H. Jonas, L. Baumgártner, M. Hollick, and B. Freisleben, "Unsupervised traffic flow classification using a neural autoencoder," in *Proceedings of the 2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, pp. 523–526, Singapore, October 2017.

[23] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: a flow sequence network for encrypted traffic classification," in *Proceedings of the IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1171–1179, Paris, France, May 2019.

[24] M. Lopez-Martin, B. Carro, A. Sánchez-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] S. Rezaei and X. Liu, "How to achieve high classification accuracy with just a few labels: a semi-supervised approach using sampled packets," 2019, http://arxiv.org/abs/1812.9761.

[26] Y. Sang, M. Tian, Y. Zhang, P. Chang, and S. Zhao, "IncreAIBMF: incremental learning for encrypted mobile application identification," in *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP)*, New York, NY, USA, September 2020.

[27] 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, 2018.

[28] X. Ren, H. Gu, and W. Wei, "Tree-RNN: tree structural recurrent neural network for network traffic classification," *Expert Systems With Applications*, vol. 167, Article ID 114363, 2021.

[29] K. Yang, L. Xu, Y. Xu, and J. Chao, "Encrypted application classification with convolutional neural network," in *Proceedings of the 2020 IFIP Networking Conference (Networking)*, pp. 499–503, Paris, France, June 2020.

[30] Y. Yang, C. Kang, G. Gou, Z. Li, and G. Xiong, "TLS/SSL encrypted traffic classification with autoencoder and convolutional neural network," in *Proceedings of the 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 362–369, Exeter, UK, June 2018.

[31] H. Zhou, Y. Wang, X. Lei, and Y. Liu, "A method of improved CNN traffic classification," in *Proceedings of the 2017 13th International Conference on Computational Intelligence and Security (CIS)*, pp. 177–181, Hong Kong, China, December 2017.

[32] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proceedings of the 2017 International Conference on Information Networking (ICOIN)*, pp. 712–717, Da Nang, Vietnam, January 2017.

[33] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "$Deep-Full-Range$: a deep learning based network encrypted traffic classification and intrusion detection framework," *IEEE Access*, vol. 7, pp. 45182–45190, 2019.

[34] H. Huang, H. Deng, J. Chen et al., "Automatic multi-task learning system for abnormal network traffic detection," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 13, no. 4, pp. 4–20, 2018.

[35] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Distiller: encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, vol. 183-184, Article ID 102985, 2021.

[36] 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.

[37] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Toward effective mobile encrypted traffic classification through deep learning," *Neurocomputing*, vol. 409, pp. 306–315, 2020.

[38] I. Akbari, M. A. Salahuddin, L. Ven et al., "A look behind the curtain: traffic classification in an increasingly encrypted web," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 5, no. 1, pp. 1–26, 2021.

[39] X. Wang, S. Chen, and J. Su, "App-Net: a hybrid neural network for encrypted mobile traffic classification," in *Proceedings of the IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 424–429, Toronto, ON, Canada, July 2020.

[40] X. Wang, S. Chen, and J. Su, "Automatic mobile app identification from encrypted traffic with hybrid neural networks," *IEEE Access*, vol. 8, pp. 182065–182077, 2020.

[41] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP)*, Rome, Italy, February 2016.