

RESEARCH

Open Access



Network traffic classification model based on attention mechanism and spatiotemporal features

Feifei Hu¹, Situo Zhang¹, Xubin Lin¹, Liu Wu¹, Niandong Liao^{2*} and Yanqi Song²

Abstract

Traffic classification is widely used in network security and network management. Early studies have mainly focused on mapping network traffic to different unencrypted applications, but little research has been done on network traffic classification of encrypted applications, especially the underlying traffic of encrypted applications. To address the above issues, this paper proposes a network encryption traffic classification model that combines attention mechanisms and spatiotemporal features. The model firstly uses the long short-term memory (LSTM) method to analyze continuous network flows and find the temporal correlation features between these network flows. Secondly, the convolutional neural network (CNN) method is used to extract the high-order spatial features of the network flow, and then, the squeeze and excitation (SE) module is used to weight and redistribute the high-order spatial features to obtain the key spatial features of the network flow. Finally, through the above three stages of training and learning, fast classification of network flows is achieved. The main advantages of this model are as follows: (1) the mapping relationship between network flow and label is automatically constructed by the model without manual intervention and decision by network features, (2) it has strong generalization ability and can quickly adapt to different network traffic datasets, and (3) it can handle encrypted applications and their underlying traffic with high accuracy. The experimental results show that the model can be applied to classify network traffic of encrypted and unencrypted applications at the same time, especially the classification accuracy of the underlying traffic of encrypted applications is improved. In most cases, the accuracy generally exceeds 90%.

Keywords Traffic classification, CNN, LSTM, Attention mechanism

1 Introduction

Network traffic classification is the process of identifying specific applications or activities by matching them with network traffic. This task is essential for network management and security [1]. In network management, network traffic classification enables the identification of

different types of network applications, allowing for the appropriate allocation of network resources [2].

Currently, some early network traffic classification methods focus mainly on two aspects: port based and deep packet inspection based [3]. Port-based methods identify network traffic by standard port numbers. However, in the current network environment, the port-based approach is beginning to decline due to the prevalence of port obfuscation and dynamic ports.

The deep packet inspection-based method has high classification accuracy for known application traffic, but it cannot identify unknown or encrypted application traffic [4]. With the development of some new technologies, machine learning, traffic behavior

*Correspondence:

Niandong Liao
Indy97@csust.edu.cn

¹ CSG Power, Dispatching Control Center, Guangzhou 510663, China

² Changsha University of Science and Technology, Changsha 410114, China

analysis, signature matching, and deep learning methods have also been applied and studied in network traffic classification [5].

Machine learning algorithms enable the classification of network traffic by training models on known categories. These models can automatically identify and classify new traffic. Common algorithms used for this purpose include decision trees, support vector machines (SVM), and Naive Bayes classifiers. They use packet features such as source and destination IP addresses, packet length, and timestamps for classification. A network traffic classification method based on Naive Bayes classification was proposed in [6]. The authors first preprocessed the dataset by removing irrelevant features and normalizing values. They then applied a feature selection algorithm based on information gain to identify the most important features in the classification task. The Naive Bayes classifier was trained on the selected features and evaluated on the test set using several metrics, including accuracy, precision, recall, and F1 scores. The study showed that Naive Bayes classification is an effective method for classifying network traffic, and that feature selection is crucial for improving the performance of the algorithm.

The traffic behavior analysis method classifies traffic by analyzing the behavioral patterns of traffic. It can detect abnormal traffic behavior and identify network attacks and unusual activities. For example, by monitoring frequent connection attempts or unusual data transfer volumes from specific IP addresses, it is possible to determine whether traffic is a malicious activity. Signature matching methods use predefined rules or patterns to match traffic. By matching it against the signature of a known network attack or malicious behavior, it can determine whether the traffic belongs to a specific type of attack. This method is commonly used in intrusion detection systems (IDS) and intrusion prevention systems (IPS).

Deep learning traffic classification is a method that uses deep neural network models to automatically classify network traffic. A deep learning-based method for packet-based network traffic classification was proposed in [7]. The authors used a convolutional neural network (CNN) to extract features from the packets and a multilayer perceptron (MLP) to classify the traffic. The method was evaluated on a real-world dataset and was shown to be more effective than other machine learning algorithms.

From the research in references [6] and [7], it can be seen that the former approach can identify certain unknown or encrypted application traffic but relies on prior knowledge. The latter method does not require manual extraction of traffic features and can automatically construct the mapping relationship between

network traffic and corresponding labels, thus eliminating the dependence on prior knowledge [8].

In recent years, the use of encryption technology has increased significantly in network communications. This is to ensure the privacy of user data. However, this has also led to a significant increase in encrypted traffic, which challenges traditional rule-based methods to effectively identify and classify it. Encryption transforms communication content into random ciphertext, making it difficult to decipher in a short period of time. As a result, load-based methods are ineffective for accurate matching and detection.

Encrypted traffic ensures secure communication, but it also creates opportunities for malicious activities. This can lead to an increase in false positives or false negatives in existing inspection methods, which poses a significant challenge for network regulators in accurately identifying and managing encrypted traffic. For example, consider an organization that relies heavily on VoIP communication for its day-to-day operations. Employees are also allowed to access video streaming platforms during their break times. However, due to limited network resources, the organization needs to prioritize VoIP traffic to ensure high call quality. However, both VoIP and video streaming traffic often use similar network ports and protocols, making it challenging to differentiate them based solely on network packet header. Existing traffic classification methods are not accurate enough to differentiate between VoIP and video streaming traffic, which leads to suboptimal QoS and potential disruptions in VoIP calls.

In addition, due to the use of encryption technology, traffic content is converted into random ciphertext, making it difficult to directly check and analyze the content. This may result in traditional content-based malicious traffic detection methods being unable to effectively decrypt and identify some normal encrypted traffic, leading to false positives. For example, Dropbox is a file synchronization and sharing service that let users upload and download files. Dropbox uses encryption to keep user data private and secure. However, this traffic behavior pattern may be similar to some malicious activities (such as massive file transfer or abnormal file types), which may result in misreported malicious traffic.

In addition, Dropbox provides its services using shared IP addresses or domain names that may be associated with other malicious activity. When cybersecurity systems detect malicious traffic based on IP addresses or domain names, they may misclassify Dropbox traffic as malicious.

Finally, Dropbox users can upload and download large amounts of data, possibly with high frequency. Such large data transfers and frequent traffic can be misinterpreted as malicious behavior, especially if traffic-

frequency-based rules are used in network security systems.

From the problems faced by the first two network applications, it can be seen that there are still some challenging problems to be solved in dealing with encrypted traffic and application traffic classification, which are mainly as follows [9–11]:

- (1) Inaccurate traffic classification: Due to the complexity and diversity of application software, traffic can vary on different devices and versions of the same application. In addition, some applications may use encryption to hide their traffic, making it difficult to classify traffic.
- (2) Misclassification: Application traffic classification is usually based on a few simple rules, such as port numbers, IP addresses, and domain names. Therefore, misclassification may occur. For example, the traffic of some applications may be very similar to that of other applications, but their functions are completely different, which can be easily misjudged.
- (3) Complex encryption algorithm: The encryption algorithm is usually very complex, which makes it difficult to classify traffic. Some encryption algorithms may also use random numbers, hash functions, and other techniques, which increases the difficulty of classification.
- (4) Traffic noise problem: Since the encrypted traffic itself is encrypted, there may be a lot of noise in the encrypted traffic, which may affect the accuracy of traffic classification.
- (5) Attacks and spoofing: Some malicious attackers may use encryption to hide their attack traffic, which poses a challenge to the classification of encrypted traffic. At the same time, the attacker may spoof other types of traffic to fool the traffic classification system.
- (6) Traffic congestion: Users may use multiple applications at the same time, and the traffic of these applications will be mixed together, making it difficult to accurately classify and record the traffic of each application.

To address the aforementioned critical issues and improve the inspection accuracy of the original methods, the following three aspects can be considered:

- (1) Update malicious traffic detection rules: Network security systems can be updated with rules based on the traffic behavior patterns and features of legitimate business traffic. This can help to avoid false positives, where legitimate traffic is mistakenly identified as malicious.

- (2) Perform comprehensive feature-based classification: Malicious traffic can be identified by considering multiple features, such as traffic behavior, data transmission patterns, IP addresses, and domain names. By considering multiple features together, it becomes possible to more accurately differentiate between normal business traffic and malicious traffic.
- (3) Use machine learning or deep learning methods: Machine learning and deep learning methods can be used to classify and identify normal business traffic. These methods learn the features and patterns of normal traffic to distinguish between normal traffic and malicious traffic.

This paper proposes an end-to-end representation learning network classification model. End-to-end representation learning is a deep learning-based approach that can directly map input data to output labels, thus avoiding the process of manual feature extraction of input data. For encryption and application software traffic classification, the solution of this paper is as follows:

- (1) Encrypted traffic classification problem: The traditional traffic classification method may need to decrypt encrypted traffic, which involves key management and privacy issues. However, the end-to-end representation learning method can directly classify encrypted traffic without decryption, thus avoiding these problems.
- (2) Application software traffic classification problem: The complexity and diversity of application software make traffic classification inaccurate. However, the end-to-end representation learning method can automatically extract useful features by learning a large amount of traffic data, so as to achieve more accurate traffic classification.
- (3) Accuracy problem: The end-to-end representation learning method can improve the accuracy and precision of classification through multi-level neural network structure and a large number of training data, which is more efficient than traditional methods.
- (4) Universality problem: The end-to-end representation learning method can deal with various types of traffic data without manual feature extraction and rule design, so it has better universality and flexibility.
- (5) Adaptive problem: Due to the constant changes of application software and encryption algorithms, traditional traffic classification methods need to be updated constantly, while end-to-end representation learning methods can update the model adap-

tively through incremental learning to adapt to new application software and encryption algorithms.

The brief steps of the end-to-end representational learning model proposed in this paper are as follows:

- (1) Sequential analysis with LSTM: LSTM is effective in capturing temporal dependencies in sequential data. It analyzes the order and timing of network packets, enabling the model to learn long-term dependencies and detect subtle patterns. This helps differentiate between normal and potential attacks, even in cases where similar ports or encryption techniques are used.
- (2) Local pattern extraction with CNN: CNNs excel at extracting local patterns and features from data. In network traffic classification, a CNN component can learn to recognize specific packet-level patterns that differentiate applications. This aids in identifying distinctive characteristics of benign Dropbox traffic, such as packet sizes, payload patterns, or protocol behavior, leading to accurate classification.
- (3) Refining feature representations with SE: The SE module enhances the model's representational power by recalibrating channel-wise features. It learns adaptive weights to focus on informative features while suppressing less relevant ones. This improves the discrimination between benign and potentially malicious traffic, reducing false-positive detections.
- (4) Combined approach: By combining LSTM, CNN, and SE in the network traffic classification process for IDS, a novel approach is introduced that leverages the strengths of each component. LSTM captures temporal dependencies, CNN extracts local patterns, and the SE module refines feature representations. This combined approach enhances accuracy and reliability in network traffic classification, specifically addressing the challenge of misclassifying benign business traffic as potential attacks and reducing false-positive detections in IDS systems.

This paper selects three datasets to evaluate the performance of the model and compares it with some methods in recent years. The results show that the model is more accurate and performs better than other methods in most cases in different classification experiments. The main contributions of this paper are as follows:

- (1) This paper proposes an end-to-end representation learning model which can automatically classify application software and encrypted network traffic. This method effectively solves the problem of

coarse and fine granularity classification of application software traffic and the difficulty of accurate classification of encrypted traffic.

- (2) This paper applies the attention mechanism and representation method to network traffic classification to get rid of the bottleneck of information processing and improve the model capability.
- (3) This paper uses three datasets to verify the effectiveness of the model. The experimental results show that the method has higher detection accuracy and stronger generalization ability than other methods.
- (4) This model can be used to identify and classify network traffic generated by different applications, even if the traffic is encrypted. This approach works by capturing underlying patterns and features within the encrypted traffic, which helps to improve classification accuracy. In the context of enterprise network environments, this model can be used by network administrators to identify and differentiate traffic generated by instant messaging tools, BitTorrent, and other applications. This helps to reduce false positives, improve network security, and ensure compliance with enterprise network policies.

The rest of this paper is organized as follows. The second section is related work, which introduces the motivation and preparation of the experiment. The third section describes the specific method. The fourth section gives the experimental results and analysis. The fifth section summarizes the paper and prospects the future research.

2 Related work

2.1 Traditional network traffic classification methods

Traditional network traffic classification refers to the method of classifying and identifying traffic by analyzing the packet features of network traffic. There has been a lot of research in this field, mainly including the following aspects:

- (1) Protocol-based classification method: This is one of the earliest network traffic classification methods, which classifies traffic by identifying the protocol identifier in the packet header. This method is simple and fast but susceptible to deception and attacks.
- (2) Port-based classification method: This method classifies traffic based on port numbers, identifying the application type by determining the source and destination port numbers of the packet. This method is also simple and fast but vulnerable to port deception attacks.

- (3) Feature extraction-based classification method: This method extracts various features of packets, such as packet size and timestamp, to classify traffic. This method requires manual selection and extraction of features, which has some subjectivity.
- (4) Traditional machine learning methods: This method uses machine learning algorithms such as SVM and neural networks to classify and identify traffic. These methods have good interpretability for classification results. In traffic classification, the interpretability of classification results is very important because important features and patterns of traffic data can be obtained by analyzing the classification results.
- (5) Deep learning-based classification method: This method uses deep learning algorithms such as CNN and recurrent neural networks (RNN) to classify and identify traffic. This method can automatically extract features and has higher accuracy and flexibility.

In the early research of network traffic classification, port-based methods were widely used in practical network business. However, these methods have lower recognition accuracy for applications with dynamic ports [12, 13]. Lim et al. fully proved that port-based methods can effectively identify applications that follow the port registration rules [14]. Most researchers now use hybrid methods that mix port-based methods with other methods to improve detection accuracy. Lu et al. proposed a hybrid method that first classifies flows into corresponding applications by packet size distribution and then groups flow into sessions by port location [15].

Another direction for improvement is deep packet inspection (DPI), a traffic detection and control technology based on the application layer. When IP, TCP, or UDP data flows pass through a DPI system, the system extracts a feature library by in-depth analysis of the packet payload. In the traffic identification process, DPI matches the network flow load to the rules in the feature library. If the match is successful, the protocol corresponding to the rule is identified. Bujlow et al. conducted a comprehensive comparison of 6 commonly used DPI tools, including 2 commercial products (PACE and NBAR) and 4 open-source tools (OpenDPI, L7-filter, nDPI, and libprotoident). The test comparison results show that the PACE commercial tool has the best detection performance among the six tools, but some open-source tools, such as nDPI and libprotoident, can also achieve very high accuracy [16].

As the rapid development of network applications, some new applications no longer follow simple port registration rules, and some are encrypted. Therefore,

both port-based and DPI-based methods cannot directly analyze network traffic [17].

In recent years, machine learning methods such as SVM, Bayesian, KNN, and neural networks have been widely applied in network traffic classification with some success. However, existing flow correlation methods based on passive flow analysis technology have problems of high storage and huge computational overhead. Hu et al. [18] proposed a novel flow correlation method based on compressed sensing-neural network. This method takes the traffic features after dimensionality reduction as the input of the convolutional neural network, extracts the correlation features through the convolutional neural network, and then uses the one-class SVM classifier to judge the correlation.

As can be seen from the above, machine learning methods have been applied to traffic classification, but their performance depends on the correct selection of traffic features and manual decision-making. This means that they cannot automatically adapt to new changes in the network. In contrast, deep learning methods generally have strong self-learning capabilities and do not require human intervention during model training. As a result, more and more researchers have begun to pay attention to and use deep learning methods to classify network flows. At present, the main popular deep learning models are convolutional neural networks (CNNs), recurrent neural networks (RNNs), and so on. These models have made great achievements in network security [19], computer vision [20], natural language processing [21, 22], speech recognition [23, 24], and other fields.

Wang et al. [9] proposed a malware traffic classification method using CNN. This method does not require manual feature engineering but directly uses raw network traffic as input to the classifier. It is the first attempt to apply representation learning to malware traffic classification, and the results of related experiments are promising. However, the authors also point out two potential limitations of the method:

- (1) The generalization ability of the method needs further verification.
- (2) The method only considers the spatial features of network traffic and ignores temporal features.

Li et al. [25] introduced RNN to network traffic classification. In this method, network datagrams are divided into several byte segments, which are then fed to the RNN for training and learning. Finally, the softmax function is applied to output the traffic type. The authors believe that this method has several advantages over traditional machine learning methods:

- (1) No prior knowledge of the target application is required.
- (2) Different protocols and multi-class operations can be handled.

To overcome the limitations of single deep learning models, researchers have begun to explore the use of multiple deep learning models in combination. For example, RNN and LSTM can be combined to analyze and refine network characteristics from multiple angles. Currently, researchers are exploring how to combine and optimize different classification methods to achieve more accurate and efficient traffic classification and identification. Additionally, researchers are also constantly exploring and researching new methods and technologies to address the challenges of classifying new applications and encrypted traffic.

2.2 Encryption and application traffic classification methods

With the widespread use of encryption and application software, network traffic classification faces new challenges. In response to these challenges, researchers have conducted a series of related work.

- (1) Encryption traffic classification: Traditional network traffic classification methods cannot accurately identify the type of encrypted traffic due to the difficulty of decrypting and analyzing encrypted traffic. Researchers have proposed encryption traffic classification methods based on traffic statistical features and machine learning algorithms, such as Hidden Markov models (HMM) and collaborative decomposition algorithms.
- (2) Application software traffic classification: The classification of application software traffic is subjective and complex. Researchers have proposed a number of methods to overcome these problems, such as host behavior, user behavior, and deep learning.
- (3) End-to-end representation learning: End-to-end representation learning is a new traffic classification method that learns the end-to-end representation of network traffic to achieve traffic classification and identification. This method can overcome the problem of manual feature selection and extraction in traditional methods and has higher accuracy and flexibility.
- (4) Network traffic classification platforms: Researchers have developed a number of network traffic classification platforms, such as OpenDPI, L7-filter, and DPI-LIB, to facilitate and accelerate traffic classification research. These platforms provide convenient traffic classification tools and datasets,

which can help researchers to conduct traffic classification research more quickly.

Network encrypted traffic classification is a technique for identifying and classifying encrypted traffic. Many researchers have attempted to solve this problem using different methods. Here are some related work introductions:

- (1) Deep learning-based encrypted traffic classification: This method employs deep learning models to automatically extract features from encrypted traffic and subsequently classify it. By utilizing models such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), the transmission characteristics of encrypted traffic can be fed as inputs to identify and categorize the traffic. Shapira et al. [26] introduce a novel method for encrypting Internet traffic. Their approach involves converting elementary stream data into images and then applying a CNN technique to identify traffic categories (e.g., browsing, chatting, video). Lotfollahi et al. [11] attempt to distinguish between encrypted and non-encrypted traffic by combining a stackable autoencoder with a CNN method. While this approach performs well on ISCX data streams, it struggles to identify subcategories of network streams when tested on Tor and YouTube data streams.
- (2) Statistical analysis-based encrypted traffic classification: This method employs statistical analysis techniques to examine the properties of encrypted traffic and extract valuable information for classifying it. For instance, Bayesian classifiers or support vector machines can be utilized to categorize encrypted traffic. Ammar Almomani proposed a system for analyzing and classifying VPN and non-VPN traffic using a new machine learning classifier called stacking ensemble learning. This approach was applied for the first time to a VPN and non-VPN attack problem. By combining predictions from multiple learning mechanisms (random forest, neural network, and support vector machine), ensemble learning was employed to enhance prediction accuracy [27].
- (3) Feature extraction-based encrypted traffic classification: This method classifies encrypted traffic by extracting and analyzing its features. For example, wavelet transform can be used to extract time-domain and frequency-domain features of encrypted traffic, and then a classifier can be employed to categorize the traffic. Okada et al. [28] investigated the impact of encryption on traffic features. They created a training dataset contain-

ing HTTP, FTP, SSH, and SMTP application protocols encrypted using PPTP and IPsec tunnels. The authors evaluated 49 traffic features and analyzed which ones had strong correlations in normal and encrypted traffic. They then used the correlated features to infer the function that transforms features between normal and encrypted traffic. Consequently, standard classifiers can be used to classify transformed traffic. The authors employed the Naive Bayesian classifier and made several modifications to validate their approach.

- (4) Rule-based encrypted traffic classification: This method uses predefined rules to determine the type and purpose of encrypted traffic. For example, the type of encrypted traffic can be ascertained by analyzing the features of the TLS handshake protocol. TLS is an encryption protocol that provides privacy for applications and is typically used to encapsulate common application layer protocols, such as HTTP and SMTP protocols. For instance, for SSL/TLS traffic, the following rules can be used to classify it:

- If the handshake message contains an RSA key exchange algorithm, the traffic type is RSA.
- If the handshake message contains an ECDHE key exchange algorithm, the traffic type is ECDHE.
- If the handshake message contains an AES encryption algorithm, the traffic type is AES.
- If the SHA digest algorithm is included in the handshake message, the traffic type is SHA.

These rules can be defined through an understanding of the TLS protocol and can be implemented through software to identify the type of encrypted traffic. Rule-based encryption traffic classification methods can quickly identify encryption traffic types, but their disadvantage is that they cannot adapt to new encryption protocols or algorithms, requiring constant updating and maintenance of rule tables.

For example, Wei et al. [29] proposed the HNNIM (hybrid neural network identification model) model to identify malicious TLS traffic. The goal is to address the issue that classical machine learning methods are heavily influenced by expert experience, resulting in suboptimal identification and classification outcomes. The HNNIM model combines plaintext information from the TLS protocol's handshake phase and the TCP protocol's header field information, reducing reliance on expert experience and effectively improving the identification and classification of malicious TLS traffic. Korczyn'ski and Duda [30] proposed a Markov-based method for detecting anomalous encrypted communication by extracting fingerprints from the payload of data packets in TLS/SSL

sessions to identify encrypted application traffic and then modeling the TLS/SSL message type sequence using a first-order Markov chain to detect anomalous encrypted communication. This method is applicable to unidirectional communication from the server to the client for a given application but requires upgrading and updating of the application and periodic updating of the fingerprints.

As new applications and services emerge, network traffic becomes more complex and diverse, making application-level traffic classification increasingly important and popular. However, existing methods for traffic classification often have limitations in achieving acceptable real-time performance.

Kyu-Seok Shim et al. [31] proposed a new method for application-level traffic classification that utilizes a sequence of payload sizes to generate unique signatures for each application. By analyzing the packet order, direction, and payload size of the first N packets in a flow, this method identifies application traffic with high accuracy and completeness rates, over 95% and 93%, respectively.

Jae-Hyun et al. [32] also used payload size sequence (PSS) signatures to classify application-level traffic. PSS signatures represent unique flow patterns for each application, which can be used to differentiate between applications. PSS signatures are generated for each application using statistical information of flows obtained from application traffic traces. This method can easily and quickly classify application traffic in real-time networks by matching the PSS signatures of new flows to those of each application.

Although some achievements have been made in application traffic classification, one of the main problems facing encrypted application traffic classification is encryption. Since the packet payload is encrypted, the traditional feature-based traffic classification method cannot be directly applied to encrypted traffic. Secondly, the characteristics of encrypted traffic and ordinary traffic are very different, so special techniques and algorithms are needed to distinguish them. In addition, as encryption technology continues to evolve and update, traffic classification methods need to be constantly updated and improved to maintain effectiveness. Finally, the number and variety of encryption applications are increasing, and the accuracy and scalability of classification methods need to be improved constantly.

End-to-end representation learning refers to the process of learning high-level features or representations from raw data using neural networks. In the field of traffic classification, end-to-end representation learning can automatically learn the characteristics of traffic without relying on manually defined characteristics.

DeepPacket and DeepFlow are traffic classification methods that utilize deep representation learning. In

[33], the authors employed a convolutional neural network (CNN) and a stacked autoencoder (SAE) to classify encrypted traffic data. They improved model performance using data augmentation and SAE learning techniques and demonstrated that DeepPacket outperformed traditional rule and feature-based methods. Similarly, in [34], the authors used CNNs and recurrent neural networks to classify data packets. They enhanced performance with data augmentation, transfer learning, and deep supervision and showed that DeepFlow outperformed traditional methods. In [35], the authors summarized various deep learning methods used in network traffic classification, including CNNs, recursive neural networks, and autoencoders. Experimental results indicated that deep learning methods outperformed traditional rule and feature-based methods and performed well in different types of traffic classification tasks.

In summary, traditional machine learning methods typically require feature extraction from network flows and classification using machine learning and rule definitions. The effectiveness of network traffic classification depends on feature extraction and rule definition and requires trustworthy public network traffic datasets for comparison. With the increasing popularity of encryption technology, analyzing encrypted network traffic effectively has become an urgent problem to solve. The complexity of network services also requires continuous exploration of how to classify different types of network flows. Although representation learning has made great progress in network traffic classification, there are still some outstanding issues to be addressed, including the following:

- (1) Classification of unknown traffic: Representation learning algorithms typically require a large amount of training data to generate effective feature representations. However, in practical networks, there may be some unknown traffic types that cannot be properly classified. Therefore, how to classify unknown traffic is still a challenge.
- (2) Classification of encrypted traffic: The widespread use of encryption communication technology has led to an increasing amount of encrypted traffic in networks. Due to the concealment of encrypted communication, encrypted traffic is difficult to be detected and recognized by traditional rule-based or feature-based classification methods. Therefore, how to effectively classify encrypted traffic is still an important issue.
- (3) Classification of variable traffic: Traffic types in networks are usually diverse and dynamically changing. For example, the traffic of the same application may change due to upgrades or updates, and even

the traffic of the same application used by different users may differ. Therefore, how to classify variable traffic is still a challenging problem.

- (4) Classification of anomalous traffic: Anomalous traffic, such as attack traffic and virus traffic, often appears in networks. The features of this traffic are usually different from normal traffic, so special classification methods are needed to identify them. However, current representation learning algorithms still face certain challenges in dealing with anomalous traffic.

3 Proposed solution

Deep learning has been proven to be very effective in network traffic classification tasks. However, the increasing diversity of network traffic and encrypted traffic demands continuous improvement and optimization of these models. In order to further improve the classification accuracy of encrypted and network application traffic, this paper proposes the use of multiple deep learning models to enhance network traffic classification. The paper also considers aspects such as dataset, feature selection, model optimization, and model fusion.

The paper emphasizes the importance of dataset quality and diversity and the need to collect valid data. For different types of traffic, appropriate end-to-end representation learning methods should be used. Using various model optimization techniques, such as adaptive learning rate, dropout, and batch normalization, can improve model performance. In addition, using multiple model fusion methods, such as voting, weighted averaging, and stacking, can further improve model performance.

The comprehensive use of these technologies and methods can effectively improve the accuracy and generalization ability of network traffic classification, especially in the area of encrypted and network application traffic classification.

In this section, a deep learning-based spatiotemporal correlation network flow classification model is proposed. The model combines the advantages of convolutional neural network (CNN) and long short-term memory (LSTM). The model framework is shown in Fig. 1.

To automatically and effectively extract and represent the spatiotemporal features of network flows, the proposed model performs the following steps:

- (1) The model utilizes LSTM to extract the temporal features of the network flow.
- (2) The model applies the squeeze-and-excitation (SE) mechanism to optimize the CNN network structure and improve its training performance.

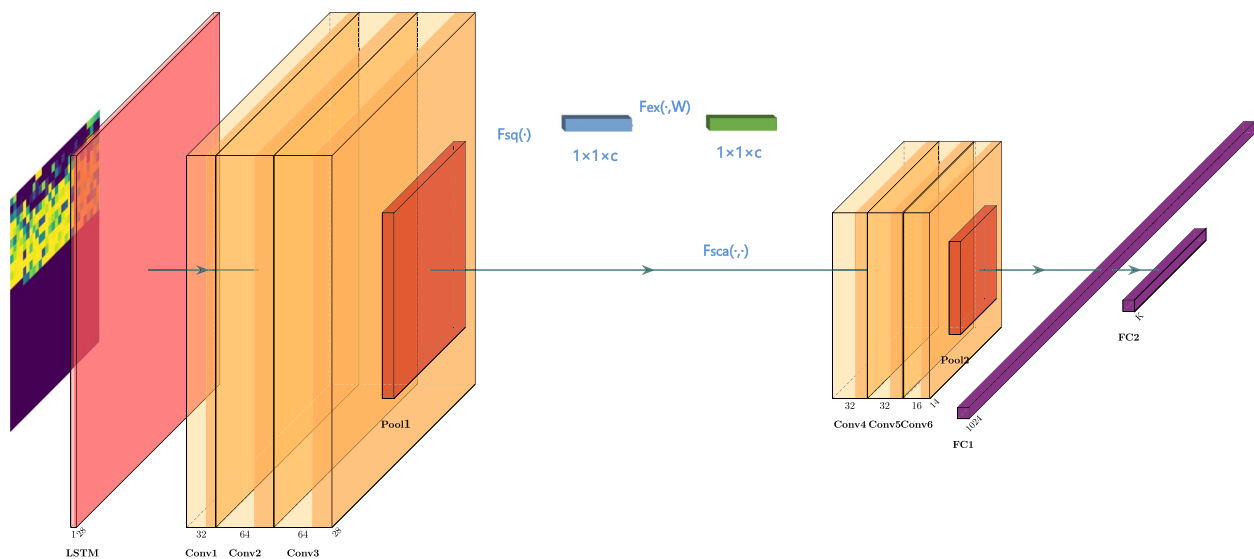


Fig. 1 Network architecture

LSTM is a type of neural network with memory function and is well-suited for processing temporal data. Given that network flow is a typical time series data, LSTM is a suitable choice for training the model.

CNN, on the other hand, has a strong ability in image feature extraction. It usually comprises three main components: convolution layer, pooling layer, and full connection layer. The convolution layer convolutes the local region of the input data with the convolution kernel, while the pooling layer reduces the dimension of the training characteristics. The fully connected layer is a traditional multilayer perceptron, often used as an output.

The CNN-SE-net is achieved by integrating the SE module with the traditional CNN network. The SE module is an attention mechanism that can be embedded in other classification or detection models. Its core idea is to learn feature weights based on the loss function through the network, so that the effective feature map has a larger weight, while the ineffective feature map has a smaller weight.

The detailed implementation of the model is shown in Sect. 3.3.

3.1 Datasets

A good dataset is a necessary condition for verifying the correctness of a method. Currently, researchers mainly rely on some well-known network attack datasets (such as KDD-CUP99 [36], NSL-KDD [37], and UNSW_NB15 [38]) to test network traffic classification methods. For example, Gao et al. [39] used the KDD-CUP99 dataset to validate their method of combining multiple Boltzmann machines and back-propagation algorithms to classify

network flows. Shone [40] and Zhang et al. [19] tested the intrusion detection systems they developed on the NSL-KDD and UNSW_NB15 datasets, respectively.

Although these datasets contain some classic examples of network attacks, some of them are outdated and cannot adapt to new network scenarios. As time goes on, network traffic and attack types and patterns are constantly changing. Therefore, evaluating intrusion detection methods using outdated datasets may lead to evaluation results that do not match the actual situation.

To address this problem, researchers can take the following measures:

- (1) Create new datasets: Researchers can create new datasets to reflect the current network attacks and traffic. This can be achieved by monitoring real-time network traffic and recording attack behavior.
- (2) Update existing datasets: Researchers can update existing datasets to reflect current network attacks and traffic. This can be achieved by adding new attack types and patterns.
- (3) Develop more generic evaluation frameworks: Researchers can develop more generic evaluation frameworks that can be applied not only to existing datasets but also to new datasets. This can be achieved by designing more flexible and scalable evaluation metrics.
- (4) Use mixed datasets: Researchers can use mixed datasets, which combine multiple datasets to reflect a wider range of attacks and traffic. This can be achieved by combining existing datasets or collecting data from different sources.

In summary, as network attacks and traffic continue to evolve, evaluating the relevance of intrusion detection methods becomes increasingly difficult. To address this issue, researchers need to take innovative approaches to create new datasets, update existing datasets, develop more generic evaluation frameworks, and use mixed datasets.

In order to verify the comprehensive ability of the proposed method in this paper, multiple different datasets were used, including three commonly used datasets: ISCX VPN-nonVPN [41], USTC-TFC2016 [42], and the YouTube dataset [43]. These three datasets contain a large amount of encrypted, unencrypted, abnormal, and normal traffic, and the proposed method was thoroughly tested from both binary and multi-class perspectives.

The ISCX VPN-nonVPN dataset consists of both VPN and non-VPN network traffic, providing a means to evaluate the performance of intrusion detection systems and network security algorithms. VPNs employ encryption and authentication to safeguard data communication, offering enhanced network security. This dataset encompasses real-world network environments and encompasses various common network protocols such as HTTP, SSH, FTP, SMTP, and DNS. Additionally, it includes instances of common network attacks like DDoS attacks, port scanning, and malware propagation.

Widely utilized in academic research and practical applications, the ISCX VPN-nonVPN dataset serves as a valuable resource for studying network attack detection, intrusion detection algorithms and tools, and testing defense performance. The Canadian Cybersecurity Institute stores this dataset in PCAP format, with network traffic categorized into 12 types based on protocol type, including chat, email, file transfer, streaming, torrent, VoIP, and more (Table 1).

The USTC-TFC2016 dataset, released by the Security Laboratory of the University of Science and Technology of China, comprises video traffic data collected from real-world scenarios covering multiple video applications and network protocols. Captured using packet capture technology between March and June 2016, the dataset includes various application scenarios spanning local networks and the Internet. It encompasses video protocols such as HTTP, RTSP, UDP, RTP, SIP, Skype, and QQ video. The USTC-TFC2016 dataset serves as a vital resource for research on video traffic analysis, video content recognition, traffic classification, network application performance evaluation, and related fields. It is widely recognized as a significant video traffic dataset. Table 1 illustrates that the dataset encompasses 20 network flows, consisting of 10 normal flows and 10 abnormal flows.

The YouTube dataset comprises 100 encrypted video streams from Chrome, with each video viewed 100 times. These videos' titles are derived from current popular topics, including news, sports, nature, and more. For example, in the directory http://www.cse.bgu.ac.il/title_fingerprinting/dataset_chrome_100/Hollyweezy/, there are 100 PCAPs, with "Hollyweezy" representing the video title. The dataset also includes some data packets with delays and packet loss for testing purposes.

The purpose of this dataset is to develop a model capable of identifying video titles associated with encrypted video traffic. In order to validate the proposed method's ability to accurately identify encrypted network flows, this publicly available dataset is employed. Due to the large number of source data samples and the space they occupy, only 10 samples are selected for testing and evaluation in this experiment, with a total PCAP stream size of 3.24 GB.

Table 1 Labels for three datasets

ISCX VPN-nonVPN		USTC-TFC2016		
Traffic	VPN_Traffic	Malware traffic	Normal traffic	YouTube
Email	VPN_Email	Cridex	BitTorrent	American_Hustle
Chat	VPN_Chat	Geodo	FaceTime	BonBon
Streaming	VPN_Streaming	htbot	FTP	Disconnect
File transfer	VPN_File transfer	Miuref	Gmail	Friends
VoIP	VPN_VoIP	Neris	MySQL	Hollyweezy
P2P	VPN_P2P	Nsis-ay	Outlook	Let_It_Go
		Shifu	Skype	Maria
		Tinba	SMB	Maroon_5_Sugar
		Virut	Weibo	Sola
		Zeus	World of Warcraft	TenYears

3.2 Preprocessing

Since the original network packets in PCAP format cannot be directly used as the input of this model, this paper refers to the literature [9] to preprocess the dataset for the PCAP packets. The specific steps include the following: traffic filtering, image generation, and IDX format conversion.

3.2.1 Step 1 (traffic filtering)

Since a session is a bidirectional network flow, it contains more abundant information than a unidirectional flow. Therefore, this paper adopts the traffic classification method based on the session mode. This step splits the PCAP-formatted raw packets into individual session-level packets and then cleans the traffic to remove empty and duplicate files that affect model training.

In this step, this paper uses the SplitCap [44] tool to split the network flow from flow to session level. SplitCap is a free and open-source PCAP file splitter. SplitCap splits a large PCAP file into multiple files based on TCP and UDP sessions, one PCAP file per session. The TCP and UDP session concepts in SplitCap are defined as bidirectional streams, i.e., all frames/packets with the same 5-tuple (source host, destination host, source port, destination port, transport protocol).

From the protocol level analysis, the traffic characteristics are mainly reflected in the application layer. For example, the STMP protocol stands for mail traffic and HTTP for browser traffic. If the relevant network features are only obtained from a single PCAP packet, the entire process of the network session cannot be accurately reflected. By using session-based traffic analysis, new implicit statistical features can be obtained, such as blocking window size, out-of-order segments, and network forward and backward flow byte sizes.

3.2.2 Step 2 (image generation)

To facilitate the processing of the CNN training model, the following steps are taken. Firstly, the cleaned files are standardized to a length of 784 bytes. If a file is longer than 784 bytes, it will be truncated, and if it is shorter, it will be padded with 0×00 bytes. Each byte in the original packet represents a pixel in the image, resulting in a conversion of 784 bytes of stream data into a 28×28 image matrix. Each image has three parameters: height (H), width (W), and channel (C).

To convert PCAPs/flows into a 784-byte image, the following steps are involved:

- (1) Obtain PCAPs/flows data: PCAPs are packet capture file formats, while flows are a data represen-

tation method based on network traffic statistics. Tools such as Wireshark and Tcpdump can be used to collect this data.

- (2) Preprocess the PCAPs/flows data: Since PCAPs/flows data is typically large, preprocessing is necessary to reduce the data volume and extract relevant information. This involves filtering out unnecessary packets, extracting packets within a specific time period, and removing irrelevant data based on specific requirements.
- (3) Feature extraction: Useful features such as packet length, source IP address, destination IP address, source port number, destination port number, and transport layer protocol type need to be extracted to convert PCAPs/flows data into an image. The selection of features depends on the specific task at hand, such as detecting malicious traffic by extracting traffic direction and duration.
- (4) Feature encoding: The extracted features need to be encoded, which can be achieved through techniques like one-hot encoding or embedding encoding. The choice of encoding method should consider its impact, such as one-hot encoding resulting in high-dimensional vectors with feature independence, while embedding encoding map features to a lower-dimensional space but may lead to information loss.
- (5) Conversion of encoded features into an image involves the following steps:
 - a) Arrange the encoded features into a one-dimensional vector according to a specific order. For example, if there are n features, each encoded with a length of m , these n features should be concatenated in order to form a one-dimensional vector of length $n \times m$.
 - b) Rearrange the one-dimensional vector into a 28×28 matrix, where the 784 values are reshaped into a matrix with 28 rows and 28 columns.
 - c) Map the values in the matrix to pixel values using techniques like linear mapping or logarithmic mapping. For instance, to map the feature value range to the integer range of 0–255, the linear mapping formula can be applied as follows: $\text{pixel_value} = (\text{feature_value} - \text{min_feature_value}) \times 255 / (\text{max_feature_value} - \text{min_feature_value})$.
- (6) Optional post-processing steps can be performed on the generated image, such as image enhancement or denoising, to improve the image quality.

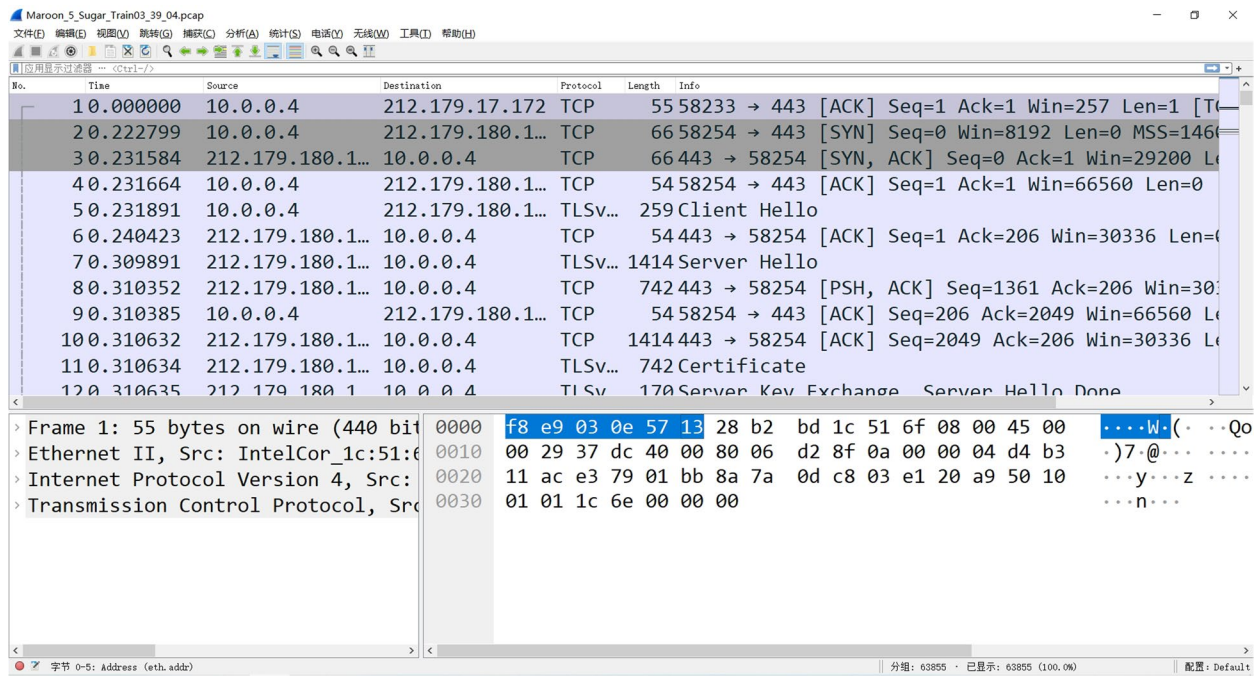


Fig. 2 Some of the original Maroon_5_Sugar flows

3.2.3 Step 3 (IDX format conversion)

The processed data is converted into the IDX file format, which serves as the input for the LSTM and CNN networks [45].

To ensure clarity in the preprocessing process, let us use an example from the YouTube dataset, specifically the flow titled “Maroon 5_Sugar.” Fig. 2 showcases a selection of original flows downloaded from the network in this study. The IDX format is commonly used for storing large multidimensional arrays or tensors efficiently. By converting the preprocessed data into IDX files, it becomes compatible with the LSTM and CNN networks, enabling further analysis and classification tasks. It is worth

noting that IDX format conversion provides a structured and standardized representation of the data, facilitating seamless integration with the chosen network models for subsequent processing and analysis.

The first PCAP network flow consists of a total of 63,855 TCP records. Visual representation of these records can be achieved using tools like Wireshark, as depicted in Fig. 3. While Wireshark allows for basic information retrieval, such as source IP address and source port, it does not provide statistical insights into network sessions.

After completing step 1, subflows in session-level PCAP format are obtained, as illustrated in Fig. 4.

Maroon_5_Sugar_Train03_39_04.pcap	2019/11/16 21:05	Wireshark capture file	64,380 KB
Maroon_5_Sugar_Train03_44_22.pcap	2019/11/16 21:05	Wireshark capture file	65,544 KB
Maroon_5_Sugar_Train03_49_39.pcap	2019/11/16 20:24	Wireshark capture file	68,377 KB
Maroon_5_Sugar_Train04_00_15.pcap	2019/11/16 21:06	Wireshark capture file	66,376 KB
Maroon_5_Sugar_Train04_05_31.pcap	2019/11/16 20:24	Wireshark capture file	65,756 KB
Maroon_5_Sugar_Train04_21_24.pcap	2019/11/16 21:05	Wireshark capture file	65,943 KB
Maroon_5_Sugar_Train04_26_41.pcap	2019/11/16 20:28	Wireshark capture file	67,923 KB
Maroon_5_Sugar_Train04_32_00.pcap	2019/11/16 21:09	Wireshark capture file	65,218 KB
Maroon_5_Sugar_Train04_37_17.pcap	2019/11/16 21:08	Wireshark capture file	65,777 KB
Maroon_5_Sugar_Train04_42_34.pcap	2019/11/16 20:28	Wireshark capture file	65,675 KB

Fig. 3 The information of Maroon_5_Sugar_Train03_39_04 flow


























 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_53295_40-78-157-239_443.pcap	2019/12/23 11:34	Wireshark captu...	4 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_57948_162-125-32-129_443.pcap	2019/12/23 11:34	Wireshark captu...	14 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58228_54-230-45-197_443.pcap	2019/12/23 11:34	Wireshark captu...	1 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58233_212-179-17-172_443.pcap	2019/12/23 11:34	Wireshark captu...	1 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58254_212-179-180-109_443.pcap	2020/7/20 16:35	Wireshark captu...	189 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58255_81-218-16-205_443.pcap	2020/7/20 16:35	Wireshark captu...	43,867 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58256_81-218-16-205_443.pcap	2020/7/20 16:35	Wireshark captu...	68,391 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58257_212-179-154-231_443.pcap	2019/12/23 11:34	Wireshark captu...	35 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58258_212-179-180-123_443.pcap	2019/12/23 11:34	Wireshark captu...	10 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58259_212-179-154-244_443.pcap	2020/7/20 16:35	Wireshark captu...	140 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58260_216-58-214-98_443.pcap	2019/12/23 11:34	Wireshark captu...	10 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58261_172-217-16-161_443.pcap	2019/12/23 11:34	Wireshark captu...	14 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58262_216-58-214-98_443.pcap	2019/12/23 11:34	Wireshark captu...	9 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58263_212-179-17-173_443.pcap	2019/12/23 11:34	Wireshark captu...	12 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58264_173-194-71-239_443.pcap	2019/12/23 11:34	Wireshark captu...	13 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58265_209-85-231-28_443.pcap	2019/12/23 11:34	Wireshark captu...	14 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58266_209-85-231-28_443.pcap	2019/12/23 11:34	Wireshark captu...	1 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58267_212-179-17-157_443.pcap	2019/12/23 11:34	Wireshark captu...	11 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58268_108-160-172-204_443.pcap	2019/12/23 11:34	Wireshark captu...	1 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58269_45-58-74-129_443.pcap	2019/12/23 11:34	Wireshark captu...	1 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58280_212-179-154-251_443.pcap	2019/12/23 11:34	Wireshark captu...	4 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58281_212-179-154-231_443.pcap	2019/12/23 11:34	Wireshark captu...	8 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58282_54-230-45-197_443.pcap	2019/12/23 11:34	Wireshark captu...	6 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_58283_216-58-213-110_443.pcap	2019/12/23 11:34	Wireshark captu...	24 KB
 Maroon_5_Sugar_Train03_39_04.pcap.TCP_10-0-0-4_59902_64-4-61-115_443.pcap	2019/12/23 11:34	Wireshark captu...	2 KB

Fig. 4 Some subflows in session-level PCAP format

Furthermore, comprehensive network communication information can be obtained by opening a session PCAP packet in Wireshark, as depicted in Fig. 5. This includes details of TCP flow establishment, data transmission, and release, along with various relevant statistical features that can be calculated.

After the processing of step 2, the network session flow is converted into a 784-byte image, which serves as the input for the CNN network. The processed images of the three datasets are displayed in Figs. 6, 7, and 8, respectively. It is evident that most of the images exhibit distinguishable texture features, while only a few share similarities, such as FTP and SMB. The traffic visualization results indicate clear distinctions between images representing different types of traffic, demonstrating the feasibility of using session flow-generated images for traffic classification.

3.3 Model

3.3.1 LSTM

Network traffic is the language of communication between computers, which is transmitted in the form of sequence in the network and contains rich time-related information. In order to comprehensively analyze the network traffic and extract its temporal correlation features, this paper adopts the LSTM method.

LSTM networks are a special type of RNN that can learn long-term dependencies. LSTM was proposed by Hochreiter and Schmidhuber in 1997 [46]. In many problems, LSTMs have achieved considerable success and are widely used. The structure of the LSTM model used in this paper is shown in Fig. 9. The long- and short-term memory network consists of several long- and short-term memory units, which are composed of linear units and a self-connection with a constant weight of 1.0. This allows a value (forward pass) or gradient (backward pass) to flow into this self-looping unit and be saved and retrieved after the desired time step.

The 784-byte data obtained from the preprocessed network stream will be converted into a matrix with a value between 0 and 255, and then, this matrix will be generated into a $28 \times 28 \times 1$ single-channel image. These images will be sent to the LSTM model to extract the time series features contained in the network flow. The internal structure of LSTM is more complex, and the core is the unit state flow shown in Fig. 9. The unit state flow is controlled and adjusted by three gate mechanisms: forgetting gate (F_t), input gate (I_t), and output gate (O_t).

The output of the previous cell h_{t-1} and the input data of the current cell x_t are entered into the forgetting gate at the same time to obtain the information retention degree of the previous hidden layer, and the value is F_t .

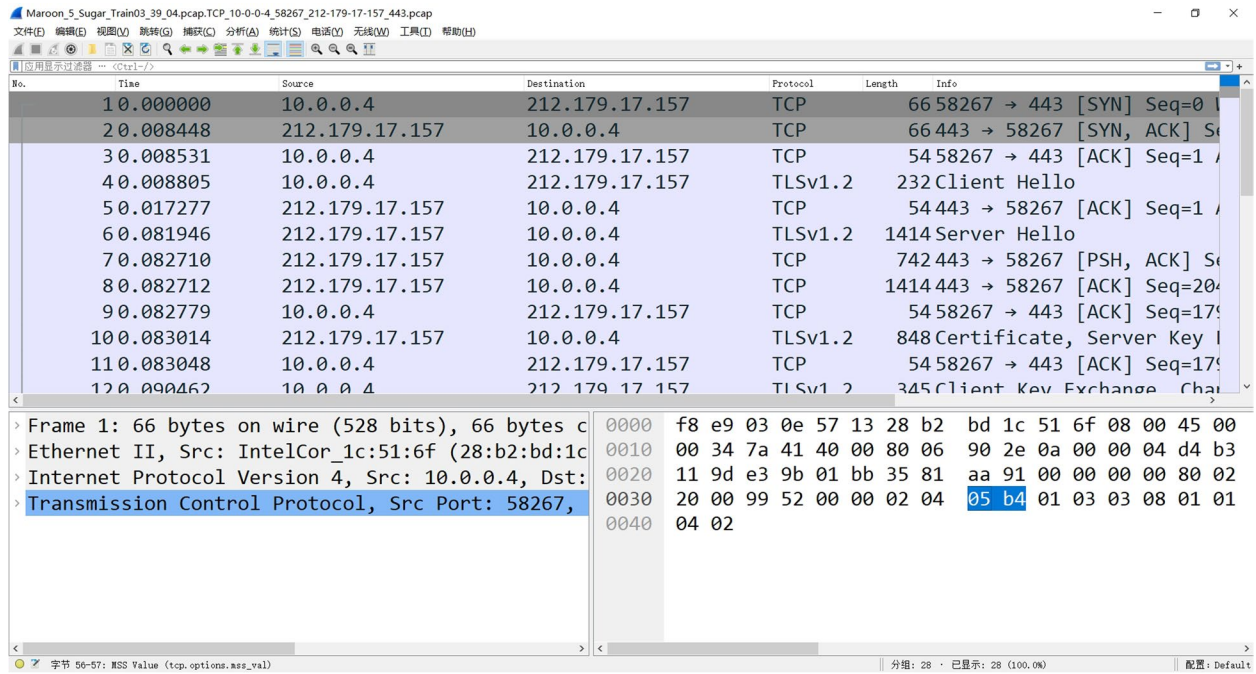


Fig. 5 A complete session network subflow

$$F_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$

where σ refers to the sigmoid function, W_f is the weight matrix of the forgetting gate, and b_f is the bias value. The input gate calculates I_t and C_t' to determine the new data and the extent to which it needs to be retained.

$$I_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$C_t' = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (3)$$

$$C_t = F_t * C_{t-1} + I_t * C_t' \quad (4)$$

In the above formula, W_i and W_c are the weight matrix of the input gate, and b_i and b_c are the bias value. Finally, according to the calculation results of the input gate and the forgetting gate, the output gate obtains the next output results h_t and O_t .

$$O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = O_t * \tanh(C_t) \quad (6)$$

where σ refers to the sigmoid function, W_o is the weight matrix of the output gate, and b_o is the bias value.

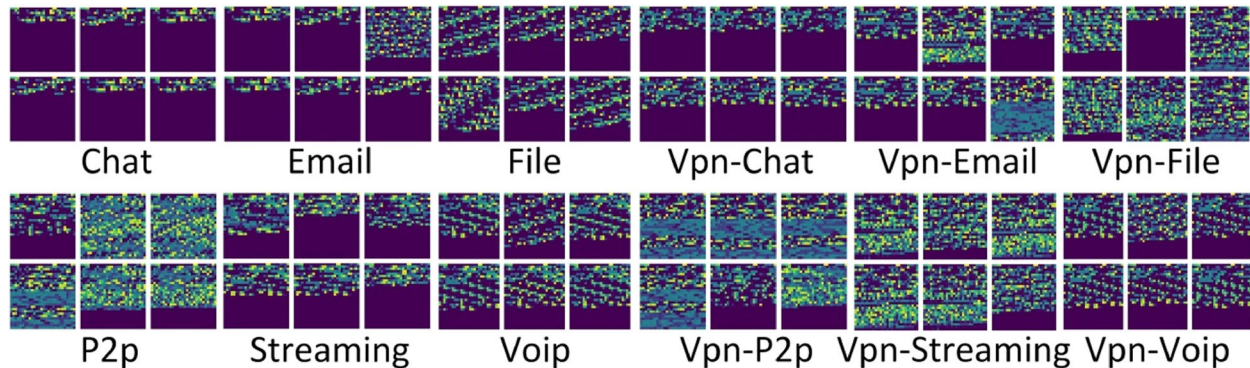


Fig. 6 Visualization of ISCX VPN-nonVPN

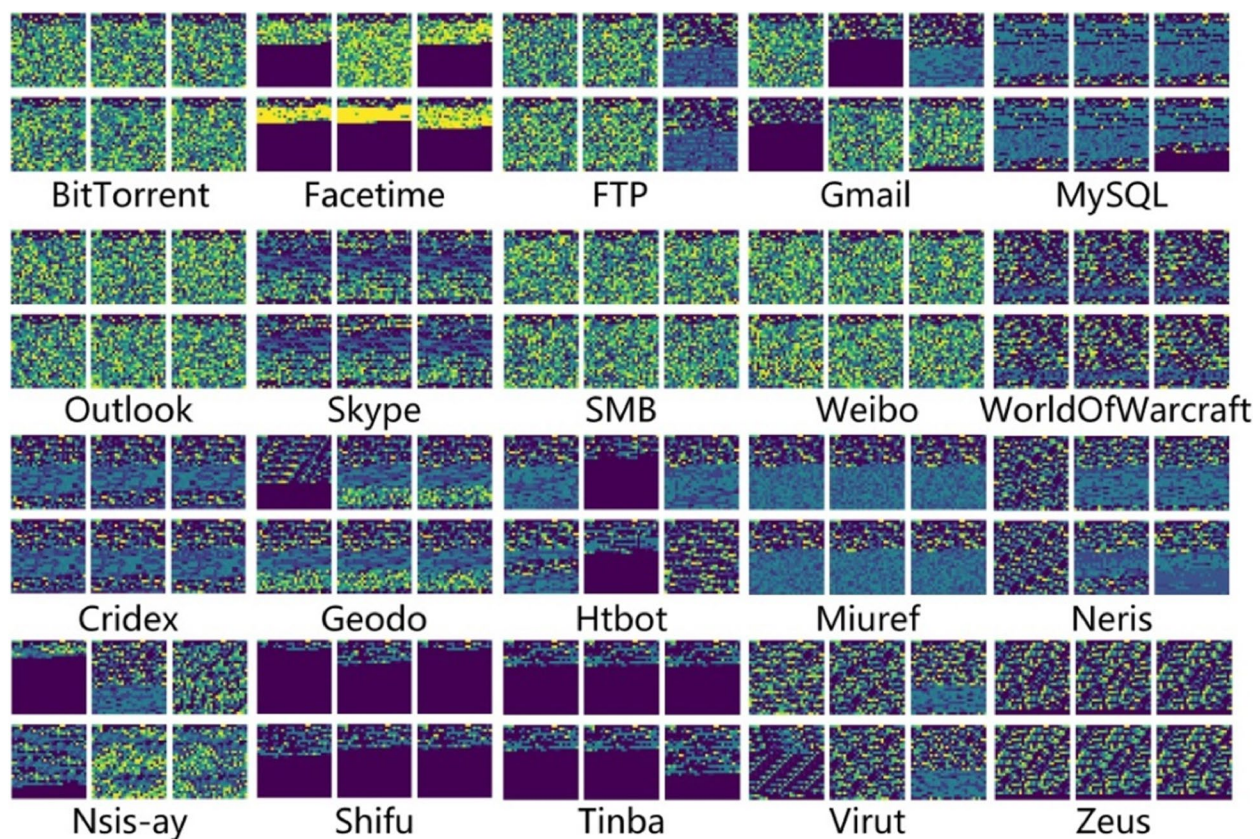


Fig. 7 Visualization of USTC-TFC2016

3.3.2 CNN-SE-net

CNN is an excellent deep learning model for image processing, capable of automatically extracting features from high-dimensional data using shared convolutional kernels, without encountering significant computational challenges.

However, researchers have identified certain limitations when employing a single CNN network. Firstly,

as the network depth increases, modifying parameters through backpropagation can lead to slower changes in parameters near the input layer. Secondly, using gradient descent algorithms may cause the training process to converge to local minima instead of the global minimum. Lastly, the pooling layer may result in the loss of valuable information and disregard the correlation between individual parts and the overall context.

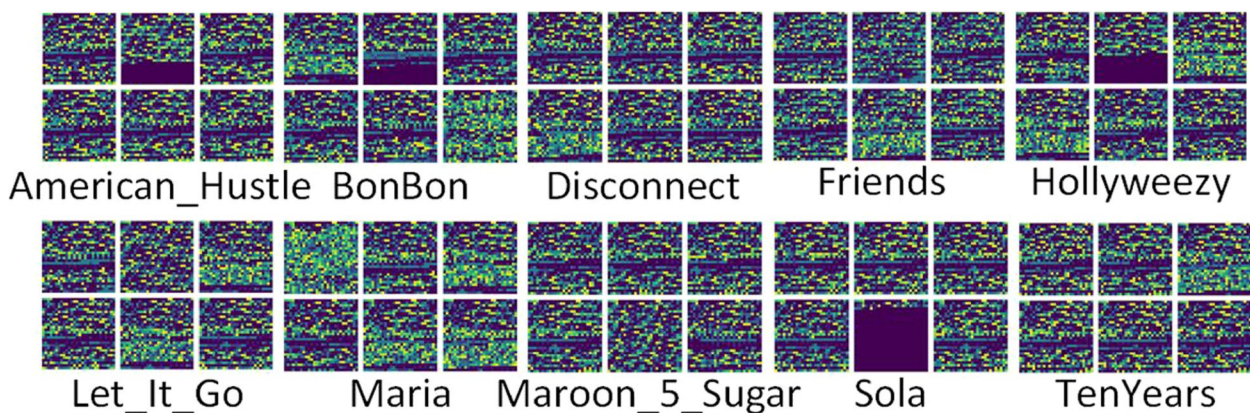


Fig. 8 Visualization of YouTube

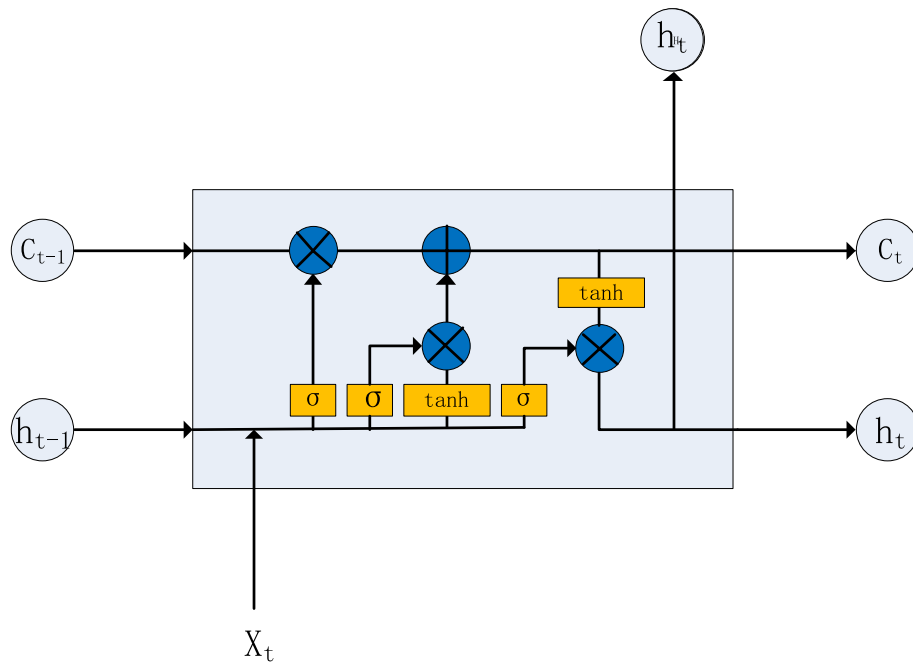


Fig. 9 The structure of the LSTM model

Consequently, it is not feasible to directly employ a single CNN model for network flow training. To enhance the accuracy and efficiency of CNN models in network traffic classification, this study proposes optimizations from two perspectives.

Firstly, during CNN model training, the large convolutional kernel's receptive field can be simulated by stacking multiple 3×3 convolutional kernels. For instance, three 3×3 convolutional kernels can be stacked to achieve a receptive field equivalent to a 7×7 convolutional kernel. This approach increases the network's depth and search space, reduces the number of model parameters, and enhances overall performance [47]. Additionally, selecting a smaller stride value during model training prevents the loss of detailed information associated with larger strides. Therefore, this study sets the model's stride parameter to 1.

Secondly, the self-attention mechanism of the SE module [48] is employed to extract spatial and channel information while recalibrating the interdependence among feature map channels. The SE module generates modulation weights based on the global information of the feature map, enhancing or suppressing different channels based on specific classification tasks.

The attention mechanism, also referred to as "neural network attention," comprises three steps: information input, calculation of attention distribution, and processing of input information based on the calculated attention distribution. By incorporating the self-attention mechanism into the CNN model, this study

effectively captures spatial and channel dependencies, leading to improved performance in network traffic classification tasks.

Let $a \in \mathbb{R}^d$ be the input vector, $X = [x_1, x_2, \dots, x_n]$ be N input samples, $q \in \mathbb{R}^k$ be the query vector or feature vector, and $Z \in [1, N]$ be the attention variable, which indicates the position of the selected information. For example, $z = i$ means the i -th input vector is selected.

The general attention mechanism is divided into soft attention and hard attention. The formulas of soft attention mechanism are generally as follow:

$$a_i = p(z = i | X, q) = \text{softmax}(s(x_i, q)) \quad (7)$$

where a_i is called attention distribution and $s(x_i, q)$ is the attention scoring function. Attention distribution a_i can be interpreted as the degree of attention of the i -th input vector for a given query q . The soft attention selection mechanism is to aggregate them.

$$\text{att}(X, q) = \sum_{i=1}^n a_i X_i \quad (8)$$

Hard attention selects information based on maximum sampling or random sampling. Among them, the formula for selecting the input information with the highest probability is as follows:

$$\text{att}(X, q) = X_j, \text{ where } j = \text{argmax}(a_i) \quad (9)$$

The specific implementation of embedding SE in CNN is shown in Fig. 10.

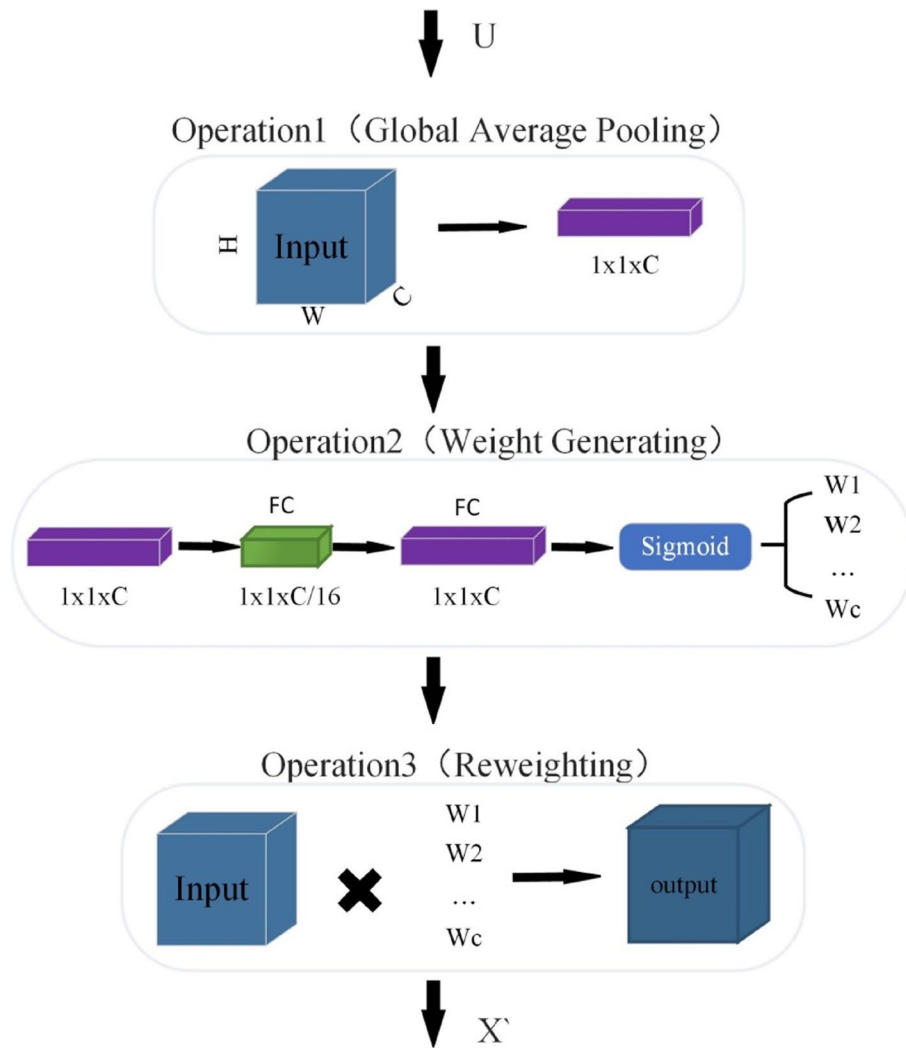


Fig. 10 SE mechanism-embedded process

After preprocessing, the original network flow is input into the LSTM module for time series analysis, and the output $X = [x_1, x_2, \dots, x_c]$ is obtained. After the convolution operation, the output is $U' = [u_1', u_2', \dots, u_c']$, $U' \in R^{H' \times W' \times C'}$.

The convolution operation is shown in formula 10.

$$U' = X * V_k, (k = 1, 2, \dots, C) \quad (10)$$

where $*$ denotes convolution and $V = [v_1, v_2, \dots, v_c]$ is C convolution kernels of 3×3 . The first three convolution layers of network use 32, 64, and 64 convolution kernels respectively, and the channel number of the feature graph is correspondingly converted to the following: 1–32–64–64.

After that, the fourth network layer is the maximum pooling layer with 2×2 and step size of 2, and the output result is $U = [u_1, u_2, \dots, u_c]$, $U \in R^{H \times W \times C}$.

SE module is embedded in the fifth layer of the model. SE module is a channel-based attention model with simple deployment and small amount of calculation [39]. In this paper, the SE module is embedded in CNN to enhance or suppress the feature channel through weight, so that the model focuses on the spatial features with greater importance.

The specific operation is further subdivided into three main steps: global average pooling $F_{sq}(U)$, weight generation $F_{ex}(w^*, W)$, and weight redistribution $F_{sca}(u_c, w_c^{\sim})$.

(1) Global average pooling

The feature map output by the maximum pooling layer has three dimensions, namely width, height, and number of channels. First, the global average pooling operation compresses the width and height directions, so that the width and height dimensions are reduced to 1×1 , but the number of channels remains unchanged.

The feature map output in this step is $w^* = [w_1^*, w_2^*, \dots, w_C^*]$, $w_1^* \in R^{1 \times 1 \times C}$. The feature map of $1 \times 1 \times C$ has a global perceptual domain, and the specific calculation of w^* is shown in formula 11:

$$w^* = F_{sq}(U) = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W u_{ij} \quad (11)$$

(2) Weight generation

Two fully connected layers are used to reduce and increase the dimension of channel C and increase the nonlinear relationship between channels. This operation produces a weight representing the importance of a set of global information $\tilde{W} = [\tilde{w}_1, \tilde{w}_2, \dots, \tilde{w}_C]$. The details are described in the formula 12.

$$w \sim = F_{ex}(w^*, W) = \sigma(g(w^*, W)) = \sigma_1(W_2 \sigma_2(W_1 W^*)) \quad (12)$$

where $W_1 \in R^{\frac{C}{r} \times C}$, $W_2 \in R^{C \times \frac{C}{r}}$, r is reduction ratio of dimensionality-reduction layer.

(3) Weight redistribution

Finally, the output $U \in R^{H \times W \times C}$ of the maximum pooling layer is multiplied by the feature channel weight $\tilde{W} = [\tilde{w}_1, \tilde{w}_2, \dots, \tilde{w}_C]$ and the output of CNN-SE-net $\tilde{X} = [\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_C]$ is obtained.

$$\tilde{x}_c = F_{sca}(u_c, \tilde{w}_c) = \tilde{w}_c u_c \quad (13)$$

It can be seen that the importance of each channel has changed after weight redistribution.

The output from the SE module undergoes another round of processing through three convolution layers, each utilizing small 3×3 convolutional kernels. These convolution layers are configured with 32, 32, and 16 convolutional kernels, respectively, resulting in a channel conversion of 32–32–16. Subsequently, the model is trained with a maximum pooling layer of 2×2 dimensions and a stride of 2. This is followed by two fully connected layers, one with a dimension of 1024 and another with a dimension matching the number of traffic classes. Finally, the softmax layer is employed to produce the ultimate classification results.

4 Experiment

4.1 Basic performance test

In this paper, three international public network traffic datasets are chosen for several experiments: ISCX VPN-nonVPN, USTC-TFC2016, and YouTube. These datasets consist of raw PCAP flow files, which are described in Sect. 3.1. After the preprocessing detailed in Sect. 3.2, each PCAP flow is transformed into a $28 \times 28 \times 1$ image. The experimental parameters are outlined in Table 2.

For evaluating the proposed detection scheme, Python, Scikit-learn, NumPy, Pandas, TensorFlow, and Keras machine learning libraries are utilized in this study. Initially, 90% of the samples are randomly selected as the training set, while the remaining 10% serve as the test set. To address potential overfitting issues, a dropout layer is introduced after the first fully connected layer. Cross-entropy and Adam are employed as the loss functions and optimizers, respectively. During the training phase, the ReLU and softmax activation functions are predominantly used.

Dropout is a regularization technique commonly employed in convolutional neural networks (CNNs) to mitigate overfitting. By randomly deactivating neurons during training, dropout reduces the complexity of the neural network and fosters independence among neurons. This, in turn, enhances the network's ability to generalize and make more robust predictions. In these experiments, a dropout value of 0.7 indicates that 70% of the neurons in each layer are randomly ignored or “dropped out” during each iteration of training. This technique effectively reduces interdependence among neurons and helps prevent overfitting, allowing the network to learn more generalized features. During

Table 2 The parameters of model

Parameter	Values
Learning rate	$10^{-2}, 10^{-3}, 10^{-4}$
Number of training (12 classifications)	3.5×10^5
Number of training (20 classifications)	6.5×10^5
Batch (12 classifications)	64
Batch (20 classifications)	256
Time stamp of LSTM	28
N_inputs (the number of input images of LSTM)	$> = 1$
N_classes (the number of network traffic classes)	$> = 2$
N_hidden_units (the number of hidden neurons of LSTM)	28
batch_size (batch size of LSTM)	64
Epochs (the number of iterations)	$> = 300$
Dropout	0.3, 0.5, 0.7

testing, all neurons are utilized to ensure optimal performance by activating the entire network for making predictions.

To assess the impact of this method and provide an objective comparison with other approaches, the experimental results in this paper are evaluated using metrics such as F1 score, recall, accuracy, and precision.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (14)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (15)$$

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

$$\text{F1 - score} = \frac{2 \times \text{precision}}{\text{Precision} + \text{recall}} \quad (17)$$

In the aforementioned formulas, TP (true positive) represents the number of successful detections of the current network traffic category. TN (true negative) refers to the number of other network traffic types that were correctly identified. FP (false positive) represents the number of other network traffic categories falsely identified as the current network traffic category. FN (false negative) denotes the number of current network traffic categories mistakenly identified as other network traffic categories.

The performance results obtained from the experiments are visualized in Figs. 11, 12, and 13. The X-axis

of these figures represents the number of training epochs for the model's training set samples. The Y-axis shows the loss value of the model training, depicted by the red curve, and the detection accuracy of the model, represented by the blue curve.

From the experimental results in these figures, it is evident that as the training epochs increase, the model's detection accuracy improves, gradually approaching 100%. Simultaneously, the loss rate decreases, progressively approaching 0%.

Furthermore, the figures demonstrate that in the selected experiments, the model generally achieves optimal results after 200 training iterations. The model exhibits a rapid convergence speed, effectively reducing the consumption of computing resources through fewer iterations of training.

To further assess the overall performance of the model and its ability to accurately classify different network traffic types, the paper conducted detection on the network applications within the three datasets. The detection results are presented in Tables 3, 4, and 5.

The results indicate that the precision, F1 score, and recall values all exhibit high performance. In Fig. 3, the average precision is 97.54%, the average F1 score is 97.61%, and the average recall is 97.72. Moreover, in Figs. 4 and 5, the average values for these three indicators also exceed 96%. These high values obtained from the detections provide strong evidence that the model proposed in this paper is capable of effectively classifying network traffic across different datasets and application types.

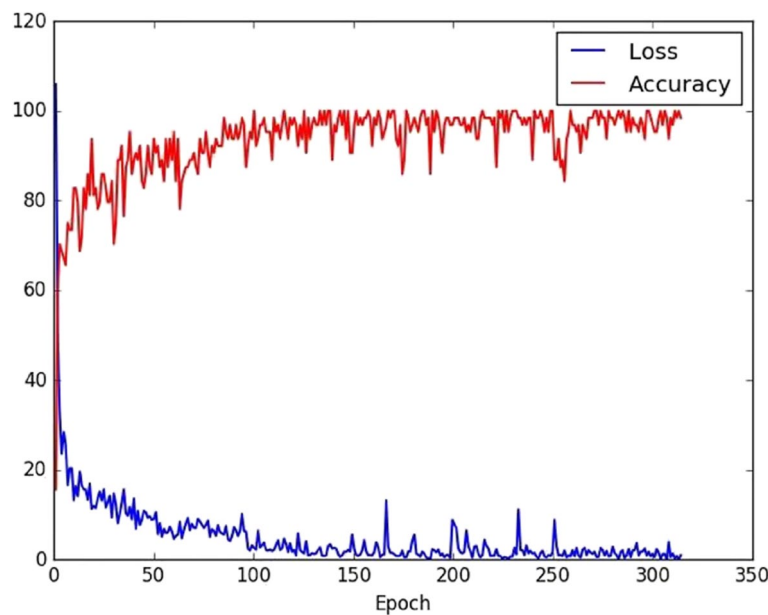


Fig. 11 Test performance on ISCX VPN-nonVPN

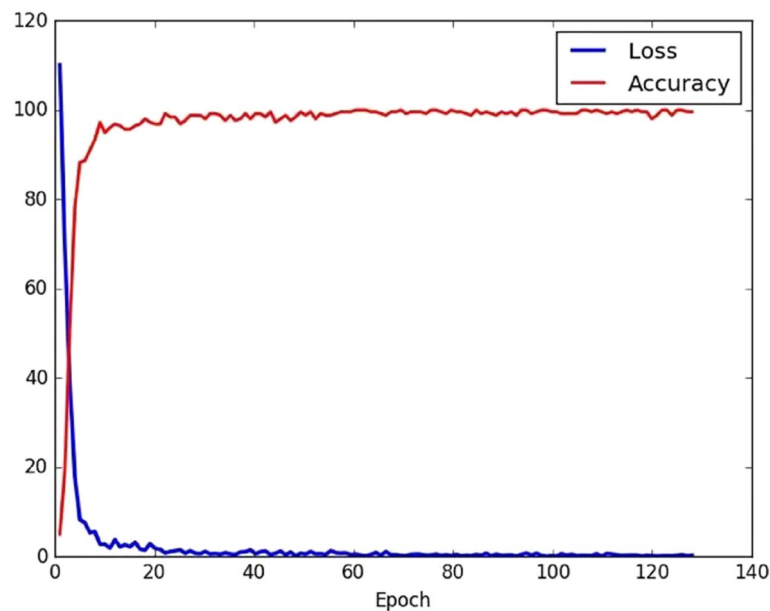


Fig. 12 Test performance on USTC-TFC2016

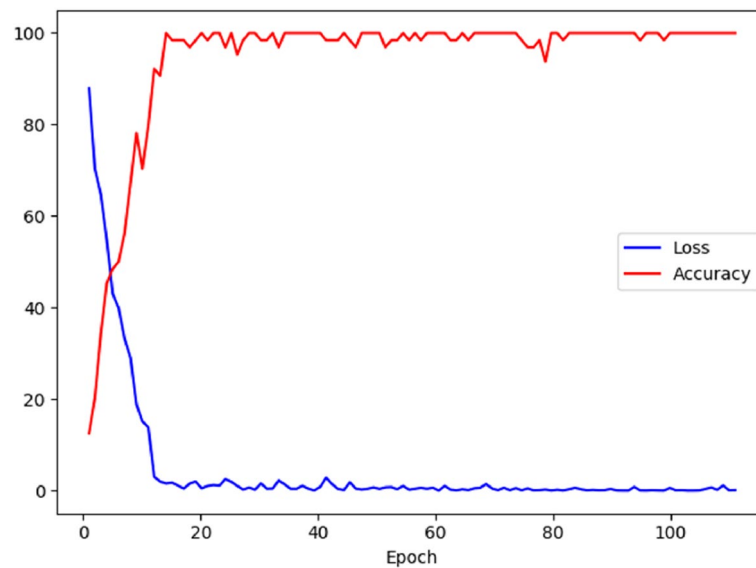


Fig. 13 Test performance on YouTube

4.2 Comparative experiment

To further validate the performance of the proposed model, the paper conducted comparative experiments with several existing methods on the first three datasets.

DraperGil et al. [10] employed the C4.5 machine learning technique to classify traffic in the ISCX VPN-nonVPN dataset. Wang et al. [1] proposed the use of multichannel LeNet-5, a variant of LeNet-5 CNN architecture, for

network traffic classification. This method demonstrated better performance in classifying VPN traffic compared to the C4.5 method. Lotfollahi et al. [11] utilized a combination of SAE and 1DCNN for network flow classification. Dubin et al. [49] employed the KNN method for flow classification.

Using the same training and test datasets, the proposed method was compared with the aforementioned

Table 3 The classification results on ISCX VPN-nonVPN

Category	Recall (10^{-2})	Precision (10^{-2})	F1 score (10^{-2})
Chat	97.01	99.84	98.40
Email	100.00	96.91	98.43
File	96.80	92.13	94.40
P2P	100.00	99.05	99.52
Streaming	98.92	99.46	99.19
VoIP	92.06	96.98	94.45
VPN_Chat	99.75	99.86	99.75
VPN_Email	93.33	90.32	91.80
VPN_File	97.00	97.98	97.49
VPN_P2P	97.92	97.92	97.92
VPN_Streaming	100.00	100.00	100.00
VPN_VoIP	99.83	100.00	99.92

Table 4 The classification results on USTC-TFC2016

Category	Recall (10^{-2})	Precision (10^{-2})	F1 score (10^{-2})
BitTorrent	100.00	99.88	99.94
FaceTime	100.00	100.00	100.00
FTP	100.00	100.00	100.00
Gmail	99.53	99.88	99.71
MySQL	100.00	100.00	100.00
Outlook	99.87	99.47	99.67
Skype	100.00	100.00	100.00
SMB	99.68	99.84	99.76
Weibo	99.85	99.85	99.85
World of Warcraft	100.00	99.87	99.93
Cridex	100.00	100.00	100.00
Geodo	100.00	100.00	100.00
htbot	99.84	100.00	99.92
Miuref	100.00	100.00	100.00
Neris	93.05	98.50	95.67
Nsis-ay	99.17	99.50	99.33
Shifu	100.00	99.86	99.95
Tinba	100.00	100.00	100.00
Virut	97.85	91.17	94.36
Zeus	100.00	100.00	100.00

literature methods, and the experimental results are presented in Tables 6, 7, and 8.

From the results in Table 6, the C4.5 method did not perform well in detecting both VPN and non-VPN flows, with a precision value lower than 85%. The other two CNN methods showed improvements in detecting VPN flows, achieving precision values exceeding 92%, but the improvement in non-VPN flow detection was not significant. The proposed method in this paper demonstrated good performance in detecting both

Table 5 The classification results on YouTube

Category	Recall (10^{-2})	Precision (10^{-2})	F1 score (10^{-2})
American_Hustle	100.00	99.98	99.94
BonBon	100.00	100.00	100.00
Disconnect	100.00	100.00	100.00
Friends	99.93	99.98	99.91
Hollyweezy	100.00	100.00	100.00
Let_It_Go	99.97	99.97	99.97
Maria	100.00	100.00	100.00
Maroon_5_Sugar	99.98	99.94	99.96
Sola	99.99	99.99	99.99
TenYears	99.95	99.95	99.95

VPN and non-VPN flows by extracting and optimizing the spatiotemporal features of network flows, with precision, recall, and F1 score values exceeding 95%.

Additionally, in Table 6, since the original literature for the C4.5 and ID CNN methods did not include the F1-score indicator, the corresponding values are indicated with “-” in Table 6.

Regarding the results in Table 7, the multichannel LeNet5 method showed strong performance in the USTC-TFC2016 dataset, achieving a detection rate of 97% for benign flows and over 98% for malware flows. Similarly, the proposed method in this paper yielded excellent results on this dataset. Although the overall indicators were slightly weaker than the former method, all three indicators for detecting the two types of flows surpassed 98%.

In the experiments conducted on the YouTube dataset (Table 8), the literature [40] primarily focused on precision values. Therefore, this paper only compared the precision value with the KNN method. The experimental results showed that the proposed method outperformed the KNN method in identifying YouTube video flows. It was capable of distinguishing between encrypted and non-encrypted flows, with a detection precision value exceeding 96%.

To evaluate the model's performance during training, tests were conducted on the ISCX VPN-nonVPN dataset. The training involved 12 different network streams, using epoch = 5000 and learning rate (LR) values ranging from 0.0001 to 0.01. The objective was to observe changes in model accuracy with varying numbers of training rounds and LR values, aiming to identify the optimal LR value.

Figure 14 illustrates the accuracy variations of the model with different LR values. It can be observed that a LR of 0.0001 led to slow convergence, as the model's accuracy failed to stabilize even after 300 epochs of

Table 6 Overall test results on the ISCX dataset

Method	VPN			Non-VPN		
	Precision (10^{-2})	Recall (10^{-2})	F1_score (10^{-2})	Precision (10^{-2})	Recall (10^{-2})	F1_score (10^{-2})
C4.5 [10]	78.2	81.3	-	84.3	79.3	-
ID CNN [1]	92	95.2	-	85.8	85.9	-
SAE + 1D CNN [11]	97.8	96.3	97	86.7	88.8	87.3
Our	99.4	99.4	99.4	97.4	97.5	96.8

Table 7 Overall test results on the USTC-TFC2016 dataset

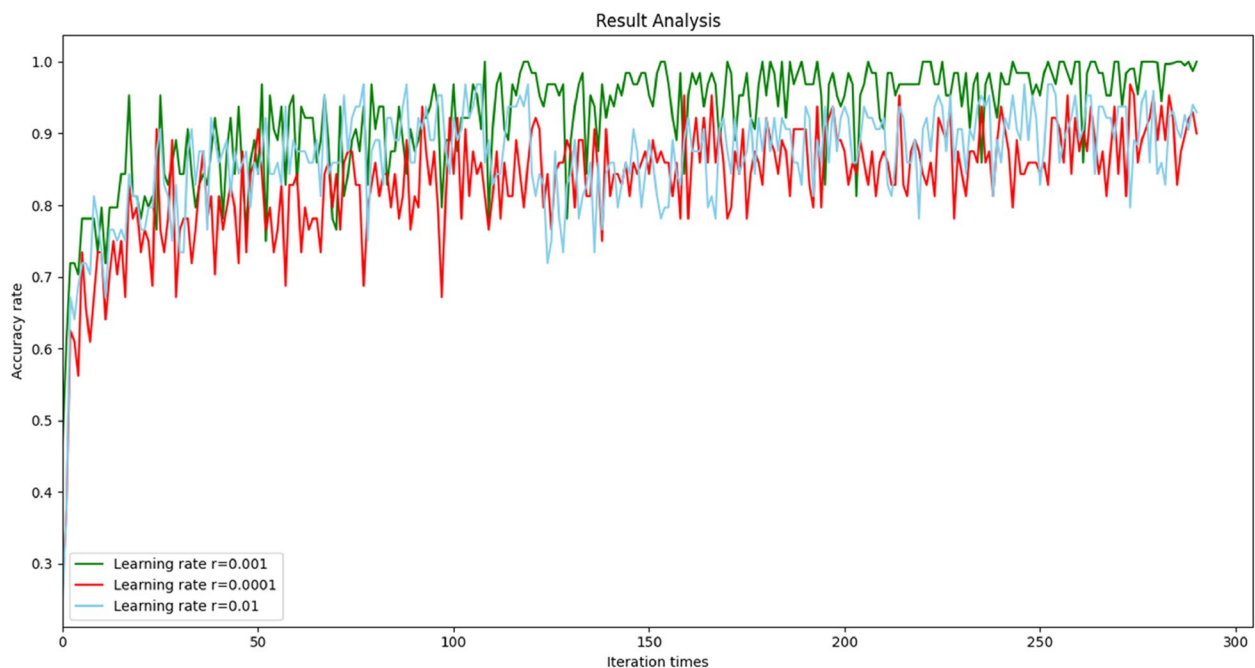
Method	Benign			Malware		
	Precision (10^{-2})	Recall (10^{-2})	F1_score (10^{-2})	Precision (10^{-2})	Recall (10^{-2})	F1_score (10^{-2})
Multichannel LeNet-5 [9]	99.7	99.7	99.7	98.6	98.7	98.6
Our	99.8	99.8	98.9	98.9	98.7	98.9

Table 8 Overall test results on the YouTube dataset

Method	K-nearest neighbor algorithm [40]	Our method
Precision (10^{-2})	95.0	99.9

training. This is due to the learning rate being a hyper-parameter that adjusts the weight degree of the network based on the loss function. A small LR value can result in longer convergence times.

In contrast, setting the LR to 0.01 caused significant oscillations in the learning rate curve, indicating that the LR was too high. This caused the gradient to swing in a region near the minimum value during training, making it difficult for the model to converge.

**Fig. 14** Accuracy of the model under different learning rates

The experiments revealed that the model achieved the fastest convergence with a learning rate of 0.001. This LR value produced a small range of oscillation in the accuracy curve and resulted in the highest accuracy rate after convergence. Therefore, a learning rate of 0.001 was selected for this study. In summary, the experiments demonstrated that the optimal LR value depends on the dataset and model architecture. Careful tuning of the LR is crucial for achieving good performance during training.

To further evaluate and compare the time-consuming performance of different models in model training, the study assesses the model's training time and convergence time. Training time refers to the duration spent by the model to train on the training set, depending on factors such as the number of iterations, model size, and experimental hardware environment. Convergence time, on the other hand, relates to the time taken for the model to reach a relatively stable performance output and is influenced by the model size and other parameters. Table 9 presents the training time required by each model (ID CNN [1], SAE+1D CNN [11], and the model proposed in this study) to complete 5000 epochs and the time taken to reach convergence based on the same test set using the ISCX dataset.

As indicated in Table 9, the model's training time for completing 5000 epochs on the ISCX dataset is 1967.76 s, which corresponds to only 93.4% of the training time of the ID CNN model and 88.8% of the training time of the SAE+1D CNN model. Moreover, the convergence time for completing 5000 epochs on the ISCX dataset is 589.17 s, representing only 81.7% of the convergence time of the ID CNN model and 85.9% of the convergence time of the SAE+1D CNN model. Tables 6 and 9 demonstrate that the proposed model achieves lower time usage compared to the 1D-CNN model with the simplest structure while maintaining an accuracy rate that is over 5% higher. Compared to the SAE+1D CNN model, the proposed model significantly reduces training time, advances convergence time, and increases the average training accuracy by more than 3%.

In summary, the proposed model demonstrates excellent performance and requires less training time than other models. However, it is important to note that actual training time and convergence time may vary due to various factors, including dataset and model parameters.

Therefore, careful consideration of these factors is crucial when selecting and optimizing models for specific applications.

5 Conclusion

Encryption stream and network application stream classification are important issues in the field of network security, which are of great significance for real-time monitoring and defending against network attacks. Aiming at the problem that existing network traffic generally has various types and it is difficult to effectively identify malicious traffic from non-encrypted and encrypted traffic, this paper proposes a deep learning traffic model based on the combination of LSTM, CNN, and SE methods. This method first eliminates the problem that some old classical machine learning methods rely too much on the accurate extraction of network traffic features. Secondly, by using the LSTM method to automatically obtain the time series features and the CNN method to obtain the spatial features of the network flow, the problems of the temporal correlation of features and the incomplete feature space are well resolved. In addition, by embedding the SE mechanism in the CNN, the correlation of the channels between different layers of the network is further analyzed to improve the accuracy of model feature selection. Judging from the results of different experiments, it fully reflects that the method proposed in this paper is indeed feasible and can basically handle the classification of different network traffic.

The model proposed in this paper can effectively enhance the capabilities of enterprises in detecting and defending against network attacks. By integrating it with IDS, IPS, and other network security systems, enterprises can achieve accurate identification and classification of network traffic. The model demonstrates higher accuracy in network traffic classification, effectively reducing false positives and false negatives. This improves the detection accuracy and reliability of IDS. The model also strengthens its feature extraction capability, allowing for a more comprehensive description of network traffic characteristics. This provides IDS with more accurate information, enabling better identification and classification of network traffic. The introduction of the SE mechanism further enhances the model's robustness by analyzing channel correlations across different layers of the network. This empowers IDS to effectively respond to variations and threats in network attacks.

With the rapid development of Internet and the gradual improvement of security awareness, there are more and more types of network traffic and encrypted traffic forms, which puts forward higher requirements for the detection rate and accuracy of the model. The future work of this

Table 9 Comparison of running efficiency of different models

Model	Training time(S)	Convergence time (S)
ID CNN [1]	2104.89	720.54
SAE+1D CNN [11]	2215.78	685.36
Our	1967.76	589.17

paper needs to continue to improve the training speed of the model and find better solutions from the aspects of network model structure and data set parallel processing.

Abbreviations

LSTM	Long short-term memory, which is a kind of time cycle neural network
CNN	Convolutional neural network, which is a kind of feed-forward neural network with convolution calculation and deep structure
SE	Squeeze and excitation, its purpose is to improve the quality of the representation generated by the network by explicitly modeling the interdependence between the channels of its convolution features
DPI	DPI is called "deep packet inspection." DPI technology adds analysis to the application layer on the basis of analyzing the packet header. It is a flow detection and control technology based on the application layer
SMTP	SMTP is a protocol that provides reliable and effective email transmission
TCP	Transmission control protocol, it is a connection-oriented, reliable, byte stream-based transmission layer communication protocol
SVM	Support vector machine, which is a generalized linear classifier that performs binary classification of data in a supervised learning method
Bayes	Bayesian classification algorithm is a classification method of statistics
RNN	Recurrent neural network, it is a type of recurrent neural network that takes sequence data as input, recursively in the evolution direction of the sequence, and all nodes (cyclic units) are connected in a chain
IDX	IDX is an image format
CNN-SE-net	CNN-SE-net refers to the SE mechanism embedded in the CNN network
BP	Back propagation, which is a multilayer feed-forward neural network trained according to the error back-propagation algorithm
C4.5	Which is an algorithm developed by Ross Quinlan for generating decision trees
VPN	Virtual private network
1D-CNN	This refers to a CNN network with a one-dimensional architecture
SAE	Stacked auto-encoders
LR	Learning rate

Acknowledgements

The authors would like to express their sincere gratitude to all the reviewers and editors for their valuable comments that have significantly contributed to improving the quality of this paper. This work has received support from the National Key R&D Program of China (No. 2020YFB0906003). Additionally, it has been partially funded by the key scientific and technological project "Research and Application of Key Technologies for Network Security Situational Awareness of Electric Power Monitoring System (No. ZDKJXM20170002)" of China Southern Power Grid Corporation. Furthermore, it has received support from the 2021 Graduate Practical Innovation and Entrepreneurial Ability Improvement Project "Research and Design of Traffic Classification Method Based on Deep Integrated Learning" (No. 6080201-000101204) from Changsha University of Science and Technology.

Authors' contributions

FH, SZ, and XL contributed to the design and implementation of the study and writing part of the paper. NL and YS conducted analysis and simulation experiments, and LW supplemented the manuscript.

Authors' information

Hu Feifei is an employee of power dispatching and control center of China Southern Power Grid Corporation. His main research interests include power system security, network situation awareness, and power data security. Email: huff@csg.cn.

Funding

There is no financial support for this study.

Availability of data and materials

The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 22 March 2021 Accepted: 1 July 2023

Published online: 12 July 2023

References

- W. Wang, M. Zhu, J. Wang, X. Zeng, Z. Yang, in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. (IEEE, Beijing, 2017), pp.43–48
- J. Zhang, X. Chen, Y. Xiang, W. Zhou, J. Wu, Robust network traffic classification. *IEEE-ACM Transactions on Networking*. **23**(4), 1257–1270 (2015)
- X. Xiao, R. Li, H. Zheng, R. Ye, A. KumarSangaiah, S. Xia, Novel dynamic multiple classification system for network traffic. *Inf. Sci.* **479**, 526–541 (2019)
- F. Michael, R. Chris, R. Eduardo, M. JeanAlexander, H. Klaus, A survey of payload based traffic classification approaches. *IEEE Communications Surveys and Tutorials*. **16**(2), 1135–1156 (2014)
- S. Rezaei, X. Liu, Deep learning for encrypted traffic classification: an overview. *IEEE Commun. Mag.* **57**(5), 76–81 (2019)
- Y. Soundharya, A. Bhanu Prasad, Network traffic classification with Naive Bayes predictions. *Int. J. Coal. Sci. Technol.* **5**(3), 51–254 (2014)
- H.-K. Lim, J.-B. Kim, J.-S. Heo, K. Kim, Y.-G. Hong, Y.-H. Han, in *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*. Packet-based network traffic classification using deep learning. (Okinawa, Japan, 2019), pp.046–051
- Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(8), 1798–1828 (2013)
- W. Wang, M. Zhu, X. Zeng, X. Ye, Y. Sheng, in *2017 International Conference on Information Networking (ICOIN)*. Malware traffic classification using convolutional neural network for representation learning (Da Nang, Vietnam, 2017), pp. 712–717.
- G. DraperGil, A.H. Lashkari, M.S.I. Mamun, A.A. Ghorbani, in *Proceedings of the 2nd international conference on information systems security and privacy*. Characterization of encrypted and VPN traffic using timerelated features. (London, the United Kingdom, 2016), pp.407–414
- M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade et al., Deep packet: a novel approach for encrypted traffic classification using deep learning. *Soft. Computing*. **24**, 1999–2012 (2020)
- Qosmos. Deep packet inspection and metadata engine, [online]. <https://www.qosmos.com/products/deep-packet-inspection-engine/>. Accessed 17 Mar 2021
- Paloalto. Paloalto networks, [online]. <https://www.paloaltonetworks.com/>. Accessed 20 Feb 2021
- Y. Lim, H. Kim, J. Jeong, C. Kim, Y. Choi, in *Proceedings of the 2010 ACM Conference on Emerging Networking Experiments and Technology*. Internet traffic classification demystified: on the sources of the discriminative power. (Philadelphia, PA, USA, 2010)
- C. Lu et al., Session level flow classification by packet size distribution and session grouping. *Comput. Netw.* **56**(1), 260–272 (2012)
- T. Bujlow, V. Carela-Espanol, P. Barlet-Ros, Independent comparison of popular dpi tools for traffic classification. *Comput. Netw.* **76**, 75–89 (2015)
- Sarah Perez. Eff: Half of web traffic is now encrypted, [online]. <https://techcrunch.com/2017/02/22/eff-half-the-web-is-now-encrypted/>. Accessed 20 Mar 2021

18. Y. Hu, H. Jin, C. Wang, Network flow association method based on compressed sensing. *Computer. Appl. Res.* **37**(S2), 245–246+ 241 (2020). (in Chinese)
19. J. Zhang, Y. Ling et al., Model of the intrusion detection system based on the integration of spatial-temporal features. *Comput. Secur.* **89**, 101681 (2020)
20. H. Zheng, J. Fu, T. Mei, J. Luo, in *The IEEE International Conference on Computer Vision (ICCV)*. Learning multi-attention convolutional neural network for fine-grained image recognition. (Venice, Italy, 2017)
21. H. Li, Deep learning for natural language processing: advantages and challenges. *Natl. Sci. Rev.* **005**(1), 24–26 (2018)
22. Y. Tom Young, H. Devamanyu, P. Soujanya, C. Erik, Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.* **13**(3), 55–75 (2018)
23. D. Wang, X. Wang, S. Lv, End-to-end mandarin speech recognition combining CNN and BLSTM. *Symmetry* **11**(5), 644 (2019)
24. J. Zhao, X. Mao, L. Chen, Speech emotion recognition using deep 1d and 2d CNN LSTM networks. *Biomed. Signal Process. Control* **47**, 312–323 (2019)
25. R. Li, X. Xiao, in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. Byte segment neural network for network traffic classification. (Banff, Canada, 2018), pp.1–10
26. T. Shapira, Y. Shavitt, in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. FlowPic: encrypted Internet traffic classification is as easy as image recognition. (Pairs, France, 2019), pp.680–687
27. A. Almomani, Classification of virtual private networks encrypted traffic using ensemble learning algorithms. *Egyptian Informatics Journal* **23**(4), 57–68 (2022)
28. Y. Okada, S. Ata, N. Nakamura, Y. Nakahira, I. Oka, in *Communications Quality and Reliability (CQR), 2011 IEEE International Workshop Technical Committee*. Application identification from encrypted traffic based on characteristic changes by encryption. (Naples, Italy, 2011), pp.1–6
29. J. Wei, R. Zheng, J. Liu, Research on malicious TLS traffic identification based on hybrid neural network. *Comput. Eng. Appl.* **57**(7), 107–114 (2021). (in Chinese)
30. M. Korkzynski, A. Duda, in *Proceedings of the 33rd IEEE Annual Conference on Computer Communications (IEEE INFOCOM)*. Markov chain fingerprinting to classify encrypted traffic. (Toronto, Canada, 2014), pp.781–789
31. K. Shim, J. Ham, B.D. Sija et al., Application traffic classification using payload size sequence signature. *Int. J. Network Manage* **27**(5), 1–17 (2017)
32. J.H. Ham, H.M. An, M.S. Kim, Application traffic classification using PSS signature. *KSII Transactions on Internet and Information Systems (TIIS)*. **8**(7), 2261–2280 (2014)
33. M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, M. Saberian, Deep packet: a novel approach for encrypted traffic classification using deep learning. *Soft Comput.* **24**(3), 1999–2012 (2020)
34. A. Lazaris, V.K. Prasanna, in *CAN 2017 - Proceedings of the 2017 Cloud-Assisted Networking Workshop*. DeepFlow: a deep learning framework for software-defined measurement. (Vilanova i la Geltrú, Barcelona, Spain, 2017), pp.43–48
35. J.W. Li, Z.S. Pan, Network traffic classification based on deep learning. *KSII Transact Internet Inform Syst.* **14**(11), 4246–4267 (2020)
36. KDD-CUP99 dataset, [online]. <https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data>. Accessed 23 Dec 2020
37. NSL-KDD dataset, [online]. <https://www.unb.ca/cic/datasets/nsl.html>. Accessed 20 Feb 2021
38. UNSW_NB15 dataset, [online]. <https://ieee-dataport.org/documents/unswnb15-dataset>. Accessed 8 June 2021
39. N. Gao, L. Gao, Q. Gao, H. Wang, in *2014 Second International Conference on Advanced Cloud and Big Data*. An intrusion detection model based on deep belief nets. (Huangshan, China, 2014), pp.247–252
40. N. Shone, T.N. Ngoc, V.D. Phai, Q. Shi, A deep learning approach to network intrusion detection. *IEEE Transact. Emerging Topics Comput Intell.* **2**(1), 41–50 (2018)
41. VPN-nonVPN dataset (ISCVPN2016), [online]. <https://www.unb.ca/cic/datasets/vpn.html>. Accessed 18 May 2021
42. USTC-TFC2016 dataset, [online]. <https://github.com/yungshenglu/USTC-TFC2016>. Accessed 8 July 2020
43. YouTube dataset, [online]. http://www.cse.bgu.ac.il/title_fingerprint/. Accessed 21 Mar 2021
44. SplitCap tool, [online]. <https://www.netresec.com/?page=SplitCap>. Accessed 13 June 2022
45. IDX file format, [online]. https://fon.hum.uva.nl/praat/manual/IDX_file_format.htm. Accessed 3 Sept 2021
46. S. Hochreiter, J. Schmidhuber, Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
47. K. Simonyan, A. Zisserman, in *The 3rd International Conference on Learning Representations (ICLR2015)*. Very deep convolutional networks for large-scale image recognition. (San Diego, CA, 2015), pp.1–14
48. J. Hu, L. She, G. Sun, in *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Squeeze-and-excitation networks. (Salt Lake City, USA, 2018), pp.7132–7141
49. R. Dubin et al., I know what you saw last minute - encrypted HTTP adaptive video streaming title classification. *IEEE Transact Inform Forensics Secur.* **12**(12), 3039–3049 (2017)

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)