Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.047918

ARTICLE

BSTFNet: An Encrypted Malicious Traffic Classification Method Integrating Global Semantic and Spatiotemporal Features

Hong Huang[1], Xingxing Zhang[1,*], Ye Lu[1], Ze Li[1] and Shaohua Zhou[2]

[1]School of Computer Science and Engineering, Sichuan University of Science & Engineering, Yibin, 644002, China

[2]School of Mathematics and Statistics, Sichuan University of Science & Engineering, Yibin, 644002, China

*Corresponding Author: Xingxing Zhang. Email: 322085406224@stu.suse.edu.cn

Received: 22 November 2023 Accepted: 29 January 2024 Published: 26 March 2024

ABSTRACT

While encryption technology safeguards the security of network communications, malicious traffic also uses encryption protocols to obscure its malicious behavior. To address the issues of traditional machine learning methods relying on expert experience and the insufficient representation capabilities of existing deep learning methods for encrypted malicious traffic, we propose an encrypted malicious traffic classification method that integrates global semantic features with local spatiotemporal features, called BERT-based Spatio-Temporal Features Network (BSTFNet). At the packet-level granularity, the model captures the global semantic features of packets through the attention mechanism of the Bidirectional Encoder Representations from Transformers (BERT) model. At the byte-level granularity, we initially employ the Bidirectional Gated Recurrent Unit (BiGRU) model to extract temporal features from bytes, followed by the utilization of the Text Convolutional Neural Network (TextCNN) model with multi-sized convolution kernels to extract local multi-receptive field spatial features. The fusion of features from both granularities serves as the ultimate multidimensional representation of malicious traffic. Our approach achieves accuracy and F1-score of 99.39% and 99.40%, respectively, on the publicly available USTC-TFC2016 dataset, and effectively reduces sample confusion within the Neris and Virut categories. The experimental results demonstrate that our method has outstanding representation and classification capabilities for encrypted malicious traffic.

KEYWORDS

Encrypted malicious traffic classification; bidirectional encoder representations from transformers; text convolutional neural network; bidirectional gated recurrent unit

1 Introduction

While information is transmitted in plaintext, it also introduces hidden dangers in the information transmission process. To mitigate the risk of eavesdropping, various applications employ the Transport Layer Security (TLS) protocol to ensure the confidentiality and reliability of the communication process. According to data sourced from the "Google Transparency Report" as of September 2021, HTTPS encrypted traffic constituted 99% of all traffic within the Chrome browser [1]. Simultaneously, encryption technology offers a means of intrusion for malicious traffic. Malicious traffic typically

denotes network activity that unlawfully infiltrates, disrupts, and captures the businesses or data of other parties. This intrusion is primarily attributed to network breaches, corporate attacks, malicious network traffic, and related program vulnerabilities. The payload information becomes invisible when traffic is encrypted, and encryption technology alters the plaintext traffic rules and parseable patterns, thereby concealing the interaction between malicious programs and servers, and evading detection by firewalls and network intrusion detection systems. Therefore, how to effectively identify malicious traffic in the context of encryption is of great significance for maintaining cyberspace security and resisting cyber attacks.

Encrypted malicious traffic classification methods can be primarily categorized into three types: rule-based matching methods, machine learning or deep learning methods based on manually extracted features, and deep learning methods based on feature self-learning. Due to the development of port obfuscation technology [2] and TLS protocol, the plaintext information of network traffic is gradually becoming ambiguous and randomized, posing a great challenge to the rule-matching method that relies on the plaintext payload information to construct fingerprints. Subsequently, researchers manually selected and extracted features based on prior knowledge, and applied machine learning algorithms that have performed well in multiple fields to identify encrypted malicious traffic. However, encrypted malicious traffic classification methods based on machine learning rely on a large number of labeled samples and multiple typical features for training, causing model performance to seriously depend on the quality of expert-designed features, with limited generalization capabilities and easily affected by human factors [3]. As deep learning has achieved great success in fields such as image recognition and sentiment analysis, researchers have begun to convert raw traffic into grayscale images and sequences and have used deep learning methods to automatically extract features from them. However, most current studies merely use a single network framework to extract features, such as Stacked Autoencoder (SAE), Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN), resulting in limited model characterisation capabilities. Subsequently, Lin et al. [4] applied the BERT model with deep feature extraction capability to traffic analysis tasks, achieving state-of-the-art performance on multiple traffic datasets. However, the model only uses the global semantic features of the packet as the final representation of traffic, ignoring the temporal and spatial features of the local bytes, resulting in limited model performance.

Currently, there are two deficiencies in the field of malicious traffic classification, the first is that the traffic classification models based on SAE, RNN, and CNN have relatively simple architectures, resulting in the inability to deeply excavate traffic features. Additionally, the BERT model with deep feature extraction capability only uses the multi-head attention mechanism to capture the global semantic features of traffic, ignoring the locally rich temporal and spatial features, resulting in too single form of traffic characterization. The above two reasons lead to poor model performance in malicious traffic classification tasks with high similarity and confusion.

To address the above problems, we need to fully explore deep features of traffic and characterize traffic from multiple feature angles. Therefore, we propose a model that fuses global semantic features and local spatiotemporal features, called BSTFNet. To extract the global features and deep semantic of traffic, we employ BERT's multi-layer transformer encoder to convert the original traffic into a high-dimensional vector representation and leverage its powerful context-learning capabilities to extract the global semantic features of traffic. Subsequently, we use the BiGRU model with excellent sequence modeling capabilities to extract deep temporal features from the high-dimensional vector representation of traffic and scaled dot product attention highlights key bytes in the traffic. Finally, we utilize the TextCNN model with multi-sized convolution kernels to extract local multi-receptive field spatial features. After the fusion of global semantic features and local spatiotemporal features,

the final representation of traffic is obtained. The proposed model offers several notable contributions, which are outlined below:

- We design a novel encrypted malicious traffic classification model BSTFNet. It innovatively integrates BERT, TextCNN, and BiGRU, which can automatically learn the global semantic features and the local spatiotemporal features of traffic from packet level and byte level respectively. The fusion of the two features can more completely characterize encrypted malicious traffic, thereby improving the classification accuracy.
- The robustness of the model was extensively evaluated on the dataset USTC-TFC2016, and comparative experiments with mature models confirmed the superiority of the proposed method. Impressively, this method achieves an accuracy of 99.39% on the public dataset USTC-TFC2016 packet-level classification, which is better than all current models.
- We conducted a detailed parameter analysis of the model and designed ablation experiments to prove the effectiveness and indispensability of each module of the BSTFNet model.

The remainder of this article is organized as follows. Section 2 summarizes the current related work on encrypted malicious traffic classification. In Section 3, we provide the overall architecture of the BSTFNet model and the detailed design of each module. In Section 4, we introduce the experimental environment, the model parameters, and the experimental results, and design comparison experiments and ablation experiments to thoroughly evaluate the model. Finally, we conclude this paper in Section 5 and present prospects.

2 Related Work

2.1 Encrypted Traffic Classification Methods

As network attacks continue to emerge, encrypted malicious traffic classification technology has received increasing attention and research from researchers [5]. According to the order in which traffic classification methods appear, they can be divided into rule-based matching methods, machine learning and deep learning methods based on manual feature extraction, and deep learning methods based on feature self-learning.

Currently, the principle of malicious traffic detection method based on rule matching is to manually establish a rule base through the prior knowledge of experts. The rule library is mainly derived from textual information, such as traffic ports, uniform resource locators (URLs), and traffic payload information. Methods based on deep packet inspection (DPI) distinguish various application types by checking the TCP/UDP port number of the traffic transport layer (such as port 23 for Telnet traffic, port 443 for HTTPS traffic, etc.) [6]. However, with the widespread use of confusion methods such as port randomization strategies [7] and camouflage strategies [8], the accuracy of port-based methods for classifying encrypted malicious traffic has dropped sharply. DPI-based methods build fingerprint information and matching rules with efficient identification effects by analyzing keywords and plaintext information in unencrypted packets during the handshake when establishing a connection and then completes the identification of encrypted traffic. Korczyński et al. [9] first extracted plaintext information from the packet payload of TLS traffic as the application fingerprint and then used a first-order Markov chain to model the information in the one-way flow from server to client. The model not only identifies encrypted applications but also detects abnormal communications between servers and clients. The disadvantage of this approach is that as the application is upgraded, the application fingerprint also needs to be updated regularly. Li et al. [10] proposed a dynamic DPI scheme, which uses dynamically symmetric cloud middleware to detect traffic packets. The middleware cannot understand

the relationship between the added rules and the data packets, effectively preventing data leakage caused by correlation analysis, and ensuring the privacy and security of DPI technology. Ning et al. [11] proposed a deep packet inspection scheme executed in a network middlebox, named PrivDPI. The DPI rules designed by Ning ensure privacy and security during the traffic detection process, and the intermediate values generated by each traffic sample can be reused in subsequent detections, greatly reducing computing delays and memory overhead. Rule-based encrypted traffic detection methods, which have the advantages of fast identification speed, high classification accuracy, and low memory overhead, can formulate corresponding matching rules according to tasks. However, its disadvantage is that the construction of rules requires a huge cost. We need to analyze massive data to build a highly discriminating rule base, and as network traffic changes, the rule base must be continuously updated to ensure the flexibility of rules.

The fundamental principle of machine learning or deep learning detection algorithms based on manual feature extraction is to manually design features through expert prior knowledge and classify them through machine learning methods such as random forest (RF), extreme gradient boosting (XGBoost), or deep learning methods such as RNN, CNN. Iliyasu et al. [12] created time series features such as interarrival time, packet length, and packet direction of consecutive packets as pseudo images and classified them using a Deep Convolutional Generative Adversarial Network (DCGAN) model. Alotaibi et al. [13] combined two machine learning models, Support Vector Machine (SVM) and Naive Bayes and used a fuzzy inference system to effectively improve the decision-making power and accuracy of the model. Anderson et al. [14] manually extracted the packet size, the packet length, and the unencrypted plaintext payload during the handshake as classification features, and then used a logistic regression model as the classification model. The results show that encrypted malicious traffic has its own unique plaintext statistical characteristics and payload information, which can be effectively identified by machine learning algorithms. Lucia et al. [15] extracted the lengths of the first 20 packets in encrypted malicious traffic as feature vectors, using positive and negative values to indicate the packet transmission direction. They achieved a higher classification accuracy and a lower false positive rate using a SVM model. Shekhawat et al. [16] extracted 38 features such as fingerprint information, cipher suite information, and connection features from the traffic logs. They analyzed these features using three machine learning algorithms (SVM, RF, and XGBoost), ultimately achieving a high classification accuracy. Torroledo et al. [17] constructed a feature engineering to extracting 40 encrypted traffic features and then used the Long Short-Term Memory (LSTM) model to obtain an accuracy of 94.87%. Shapira et al. [18] converted the packet payload size distribution of flows into square grayscale images and then classified them using a 2D-CNN model. In most cases, machine learning based on artificial features is oriented to different application scenarios, so it is necessary to select appropriate features and algorithms based on domain knowledge to train model. The main factor determining model quality is feature engineering, which directly affects the classification accuracy and generalization capability [19]. The manual feature selection process not only heavily relies on expert experience but also extracts features with limited dimensions, resulting in a model that suffers from the disadvantages of overfitting and poor generalization ability.

Deep learning methods based on feature self-learning make up for the shortcomings of machine learning methods. Its principle is to use the nonlinear modeling ability of neural networks to extract the feature vector of traffic and then fit traffic through representation learning. As deep learning technology has achieved great success in fields such as natural language processing and image recognition in recent years, researchers have also begun to use end-to-end deep learning methods in encrypted traffic classification and malicious traffic identification tasks. Wang et al. [20] applied the end-to-end representation learning method to the encrypted traffic classification task for the first

time. He first converted traffic into grayscale images and then used a convolutional neural network to classify them. Through comparative experiments between one-dimensional convolutional neural network (1D-CNN) and two-dimensional convolutional neural network (2D-CNN), Wang verified that the nature of traffic is a one-dimensional time series, and 1D-CNN has more advantages in extracting the characteristics of traffic. Lotfollahi et al. [21] integrated the SAE model and the CNN model to develop a "Deep Packet" framework for traffic classification. Among them, the SAE model completed the binary classification task of normal and abnormal traffic, and the CNN model achieved fine-grained classification of applications on this basis. The results proved that the combination of multiple deep-learning models can effectively improve classification accuracy. Hwang et al. [22] truncated or padded the first N packets in the session stream to length L and then converted them into grayscale images of size $N \times L$. Finally, a model consisting of an autoencoder and a one-dimensional convolutional network was used for malicious traffic detection. Chen et al. [23] used ArcMargin to optimize the feature extraction layer of the CNN model, which effectively improved the resolution ability of the CNN model and enabled the clustering of unknown traffic. In a complex network environment, relying solely on the spatial features of traffic will lead to a serious decline in classification accuracy. Due to the time series attributes of network traffic, RNNs, known for their powerful sequence feature extraction capabilities, are also employed together with CNNs in traffic classification tasks. Liu et al. [24] proposed an FS-Net model that automatically extracts sequence features from original traffic packets. They utilized a multi-layer encoding and decoding structure to mine high-dimensional sequence features from encrypted traffic and used a reconstruction mechanism to enhance the discriminability and effectiveness of the features. Lin et al. [25] proposed an encrypted traffic classification model TSCRNN, which combines the CNN model and the bidirectional long short-term memory (BiLSTM) model. The CNN model extracts the spatial features of traffic, and the BiLSTM model extracts the temporal features of traffic. Zou et al. [26] utilized the CNN model and the LSTM model to learn the low-level spatial features and high-level temporal features of network traffic, respectively, which ultimately improved the classification accuracy effectively. Liu et al. [27] built the bidirectional gated recurrent unit attention (BGRUA) model for encrypted traffic classification, which uses BiGRU's bidirectional modeling capability to extract bidirectional time series features of trafficand then uses the attention mechanism to increase the weight ratio of important features. The experimental results showed that the BGRUA model has achieved good results in four indicators: accuracy, precision, recall, and F1-score. However, methods based on deep learning rely on a large amount of labeled traffic data. When the data is insufficient, the performance of the model will be greatly affected [4], and there are shortcomings in single and shallow feature expressions.

With the widespread application of the ransformer model in the field of natural language processing, researchers have introduced the self-attention mechanism of the Transformer model into traffic analysis tasks. Wang et al. [28] combined the transformer model with powerful sequence modeling capabilities and the CNN model with excellent spatial feature extraction capabilities to design a hybrid deep learning model that can efficiently identify malicious traffic. The experimental results showed that the hybrid model has better results in various indicators than the individual model. Subsequently, Google released the pre-trained language representation model BERT [29] based on a bidirectional Transformer. He et al. [30] introduced the BERT model into traffic classification, proposing a model that utilizes BERT's multi-head attention mechanism to learn traffic context relationships. When performing downstream classification tasks, they used a small amount of data to fine-tune the BERT pre-trained model to fit the classification task. The experiments were conducted on multiple publicly available encrypted traffic datasets, and the results showed that the proposed model outperformed other baseline models. Lin et al. [4] employed a multi-layer bidirectional transformer

architecture to capture contextual relationships of traffic. They used a large-scale unlabeled dataset to build a pre-trained model, fine-tuned the pre-trained model for specific tasks, and ultimately achieved robust performance on multiple encrypted traffic datasets. Shi et al. [31] captured byte-level and packet-level features through the BERT module and the CNN module and then concatenated them as the final features. The experimental results show that it can effectively enhance the performance of the traffic classification model. Table 1 summarizes the main research results, including the traffic features and analysis methods used.

Table 1: Summary of existing mainstream methods

Reference	Years	Features	Methods
[9]	2014	Payload information	Markov
[10]	2022		DPI
[11]	2019		DPI
[12]	2020	Artificial statistical features	GAN
[13]	2019		RF
[14]	2018		Logistic
[15]	2019		SVM
[16]	2019		SVM, RF, XGBoost
[17]	2018		LSTM
[18]	2020		2D-CNN
[20]	2017	Automatically extract spatial features	CNN
[21]	2020		SAE, CNN
[22]	2020		SAE, 1D-CNN
[23]	2020		CNN
[24]	2019	Automatic extraction of temporal features	RNN
[27]	2020		BiGRU
[25]	2021	Automatically extract spatiotemporal features	CNN, BiLSTM
[26]	2018		CNN, LSTM
[4]	2022	Automatically extract global semantic features	BERT
[31]	2023		BERT
[32]	2023		GPT
[28]	2021	Automatically extract global semantic and spatial features	Transformer, CNN
[30]	2023		BERT, CNN

From the table, two shortcomings of the current research can be seen. Firstly, the use of shallow models such as CNN and RNN leads to a lack of depth in feature extraction. The second is that despite the use of the deep feature extraction model BERT, it does not make full use of the local spatiotemporal features of traffic. The BSTFNet we proposed can effectively solve these problems.

2.2 Pretrained Models

Pre-trained models are trained on large-scale unlabeled datasets to learn universal semantic representations. These general features can be applied to a variety of tasks because they reflect

general patterns in the data. Subsequently, by fine-tuning a model on specific tasks, a semantic representation of traffic can be obtained. Payload encoding representation from transformer (PERT) [30] initially performed service identification on encrypted traffic using the open-source A Lite BERT (ALBERT) pre-trained model. The drawback of the PERT model is the absence of pre-training tasks specifically designed for encrypted traffic, which constrained its generalization capability. ET-BERT [4] constructed a pre-trained model specifically designed for traffic classification by employing two self-supervised pre-training tasks: the masked BURST model and the same-source BURST prediction. In comparison to the PERT model, the ET-BERT model exhibited more robust performance and superior generalization across multiple tasks. Meng et al. [32] proposed a generative pre-training model NetGPT for traffic understanding and generation tasks, which encodes multi-modal network traffic into a common semantic space and obtains basic pre-training models that support different tasks. However, the above models only use the global semantic representation as traffic features and lack attention to the local spatiotemporal features of traffic, resulting in overly monotonous representation of traffic in the models.

3 Methodology

The encrypted malicious traffic classification method proposed in this article mainly includes three steps: data preprocessing, malicious traffic characterization, and malicious traffic classification. The model structure is shown in Fig. 1. Firstly, we perform data preprocessing operations on the raw Pcap file, including traffic segmentation, data cleaning, length standardization, format conversion, and sequence generation, with the entire process being carried out using the Datagram2Token tool [4]. Subsequently, The preprocessed data is used as the input of the model through token embedding, position embedding and segment embedding, and BERT's attention mechanism is used to obtain a vector representation that incorporates global semantic information. Byte-level vectors ($h^N_1, \ldots, h^N_{[SEP]}$) and sentence-level vector $h^N_{[CLS]}$ are the outputs of this process, with the vector $h^N_{[CLS]}$ corresponding to the global semantic feature of the data packet. We use the BiGRU model to extract bidirectional timing features of byte-level vectors and then employ the scaled dot-product attention to improve the feature expression ability of important bytes. We splice timing feature vectors and byte-level vectors into timing extension vectors, which has the advantage of enhancing the ability to express timing features while avoiding the loss of original features caused by module series connection. Then the TextCNN model with convolution kernels of different sizes is employed to extract local spatiotemporal features. Global semantic features and local spatiotemporal features are fused to obtain the final representation of traffic, and finally, the Softmax function is used to achieve fine-grained classification of encrypted malicious traffic.

3.1 Data Preprocessing Layer

To minimize the potential impact of noise and information redundancy in the original dataset on the model performance, and to ensure that data samples align with the input format of the model, we utilize the Datagram2Token tool for dataset preprocessing. First, the original PCAP files are segmented at packet granularity. Subsequently, packets with a size smaller than 80 bytes are removed, as they do not contain any meaningful payload information. Simultaneously, traffic irrelevant to the transmission content, such as ARP, ICMP, and DHCP protocol packets, is also eliminated. As strong identifiers (such as ports and IP addresses) do not carry valuable information for distinguishing samples, focusing on these details during model learning can potentially lead to overfitting. To avoid bias and interference, we removed the IP header, the protocol ports of the TCP header, and Ethernet headers [33]. Subsequently, the packet data is read in hexadecimal, and the hexadecimal traffic

sequence is integrated into double bytes, with each unit consisting of four hexadecimal digits (two bytes). The processed data is shown in Table 2. To mitigate the bias introduced by data imbalance, we randomly selected 5000 samples from each category (with 2110 samples in the Shifu class), and finally divided the dataset into a training set, validation set, and test set according to the ratio of 8:1:1.

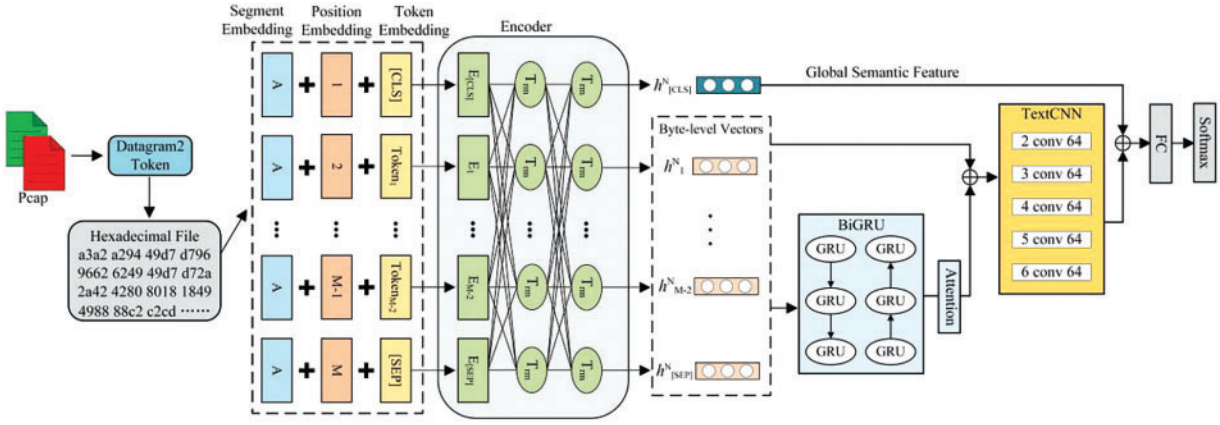


Figure 1: BSTFNet model architecture

Table 2: Preprocessed data format

Label	Hexadecimal sequence
14	feb5 b541 41d5 d558 58ed edf4 f426 26d2 d251 5180 8018 1849 4988 88d2 d2ac ac00 0000 0001 0101 0108 080a 0a1a 1a61 610c 0cfd fd08 0834 34f2 f2cf cf99 995a 5a32 32f2 f2f6 f64b 4bb4 …
9	0050 5087 8756 56cf cf38 38d7 d706 0634 3463 6350 5018 18fa faf0 f05e 5e12 1200 0000 0047 4745 4554 5420 202f 2f67 6765 6574 746a 6a73 736f 6f6e 6e3f 3f34 3470 7071 7137 3766 …
1	1f90 90b9 b9f9 f9a0 a002 0275 7537 371a 1a02 0250 5018 1850 5000 005a 5ac6 c600 0000 0041 4111 1112 121c 1c19 1918 180f 0f43 4341 411a 1a18 1809 0922 2210 1012 1219 1908 0811 …

3.2 Malicious Traffic Representation Layer

3.2.1 BERT-Based Global Semantic Feature Extraction Module

ET-BERT captures byte-level and BURST-level context relationships through two self-supervised pre-training tasks: a network traffic-specific masked BURST model and homologous BURST prediction. It leverages large-scale unlabeled data for pre-training, refining a universal packet semantic representation. In this paper, we construct a global semantic feature extraction module using the ET-BERT pre-trained model. The module consists of 12 Transformer blocks, and the representation dimension for each input token is set to 768. This design equips it with powerful global semantic representation capabilities. The BERT model includes two stages: (1) word embedding stage: including token embedding, segment embedding, and position embedding; (2) encoding stage: in this stage, the previously mentioned token vectors, sentence vectors, and position vectors are linearly summed to generate composite embedding representations, which serve as the input to the encoder layers. Then,

we utilize the multi-head attention mechanism to capture a vector representation that integrates global semantic information.

Token embedding converts each token in the sequence into a vector by querying the dictionary table. The range of each token is from 0 to 65535, and the maximum value of dictionary |V| is 65536. At the same time, during the word embedding process, the input sequence will be added with two special tokens [CLS] and [SEP]. Token [CLS] is located at the beginning of the sequence, and token [SEP] is placed at the end of the sequence. When the input length is less than the model length requirement, the token [PAD] is filled at the end of the sequence. When the input sequence length exceeds the model length requirement, a truncation operation is performed.

Position embedding numbers the position of each token in the sequence. Since the transmission of traffic data is closely related to the order, position embedding needs to be used to ensure that the model pays attention to the temporal relationship between tokens through relative positions. Therefore, each input token is assigned an H-dimensional vector to represent its position information in the sequence, where H is set to 768.

Segment embedding is a marker used to distinguish two sentences in an input sentence pair. It classifies two sentences based on whether they are semantically similar so that the input sentences can be simply connected and input into the model. It is mainly used in the pre-training stage, and this article does not cover such work.

In the encoding layer, the BERT model uses a multi-layer bidirectional transformer encoder to encode the input vectors. Each T_{rm} corresponds to a unit encoder, ($E_{[CLS]}$, E_1,..., $E_{[SEP]}$) are the input vectors of the BERT module, and ($h^N_{[CLS]}$, h^N_1, . . . , $h^N_{[SEP]}$) are the output vectors of the module. We use $h^N_{[CLS]}$ as the global semantic features of traffic, because the [CLS] token itself does not contain any semantic information and can more fairly integrate the semantic information of other tokens in the sequence, thereby better representing the global features of traffic. The BERT model architecture is a multi-layer bidirectional Transformer encoder, whose structure is shown in Fig. 2.

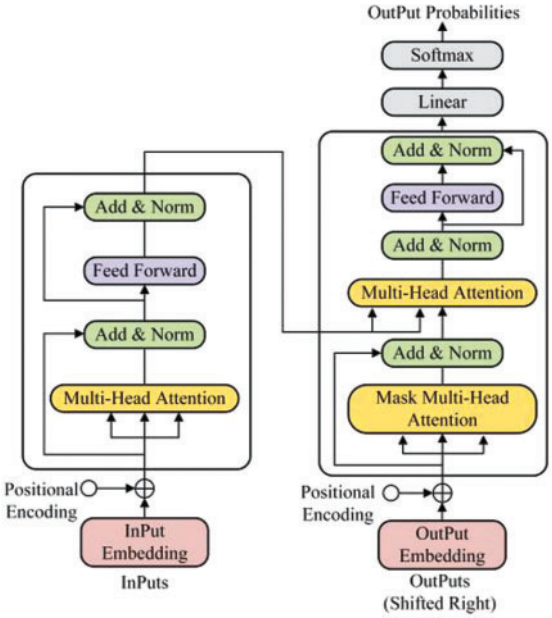


Figure 2: BERT model architecture

The core of the transformer is the self-attention mechanism, which can learn the relationship between any two tokens in the sequence. Its principle is to construct three matrices Q, K, and V to query the relationship between the current token and other tokens in the context and update the feature vector. The updated feature vector contains information about the entire sequence. The calculation process of the self-attention mechanism can be expressed as Eq. (1):

$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V \tag{1}$$

where Q is the query matrix, K is the key matrix, V is the value matrix, and d_k is the matrix dimension of Q and K. BERT enhances the expression ability of the model by using multi-head self-attention. Multi-head self-attention linearly projects the query, key, and value matrices h times, and obtains the attention weights by executing scaled dot-product attention in parallel. Finally, the h attention weights are multiplied with the value matrix, and then merged to generate a new representation. The multi-head self-attention calculation process can be expressed as Eqs. (2) and (3):

$$h_i^{\text{head}} = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \tag{2}$$

$$\text{MultiHead}\,(Q, K, V) = \text{Concat}\left(h_1^{\text{head}}, h_2^{\text{head}}, \cdots, h_h^{\text{head}}\right)W^0 \tag{3}$$

where h represents the number of attention heads, and W_i^Q, W_i^K, W_i^V are the weight matrices for the h_i^{head}.

3.2.2 Temporal Feature Extraction Module

In recent years, methods combining BERT and LSTM have performed well in natural language processing, such as sentiment analysis [34], and sarcasm detection [35]. Studies have shown that using models such as LSTM on top of the BERT model can mine deeper temporal features of sequences. Inspired by this, we use BiGRU after the BERT model to extract deep temporal features of traffic. Compared with LSTM, GRU has a simpler structure, fewer parameters, and lower complexity. GRU comprises two gate structures: the reset gate and the update gate. The reset gate determines the extent to which information from the previous time step is ignored, and the larger the reset gate, the less information is ignored. The update gate controls the degree to which information from the previous time step is brought into the current state. The larger the value of the update gate, the more status information is brought into the previous moment [36]. The structure of the GRU unit is shown in Fig. 3a, x and h represent input and output respectively, r and z represent reset gate and update gate respectively. The update gate outputs a value z_t between 0 and 1. z_t determines the extent to which the information h_{t-1} at the previous moment is passed to the next state. The reset gate controls the importance of h_{t-1} to the result h_t. Then based on the update gate, the memory information g_t at the current moment is generated, and the output h_t is calculated. The GRU calculation process can be expressed as Eqs. (4) to (7):

$$z_t = \sigma\left(W_z x_t \oplus U_z h_{t-1}\right) \tag{4}$$

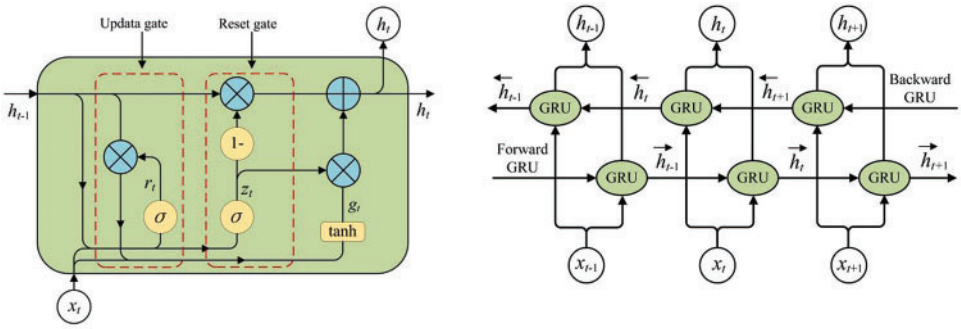
$$r_t = \sigma\left(W_r x_t \oplus U_r h_{t-1}\right) \tag{5}$$

$$g_t = \tanh\left(W_g x_t \oplus U_g(r_t \otimes h_{t-1}\right) \tag{6}$$

$$h_t = (1 - z_t) \otimes h_{t-1} \oplus (z_t \otimes g_t) \tag{7}$$

where σ represents the sigmoid activation function, W_z, W_r, W_g, U_z, U_r, U_g represents the weight parameter, $\oplus$ represents the elementary addition operation, $\otimes$ represents the elementary multiplication operation, and tanh represents the hyperbolic tangent activation function of the candidate hidden state.

A unidirectional GRU can only establish connections between the current input and historical information, failing to capture the impact of future input on the current input. However, there is a temporal relationship between bytes and their adjacent bytes in traffic data. Therefore, we choose the BiGRU model to extract temporal features. The BiGRU model consists of both a forward and a backward GRU, and their complementary nature enables the model to establish connections between preceding and succeeding byte states. The structure of the BiGRU model is depicted in Fig. 3b.



(a) GRU model architecture (b) BiGRU model architecture

Figure 3: GRU model architecture

The BiGRU module receives the output vector of the BERT module as input and deeply mines the time series features in it. The calculation process is shown in Eqs. (8) to (10):

$$\vec{h}_t = GRU\left(x_t, \vec{h}_{t-1}\right) \tag{8}$$

$$\overleftarrow{h}_t = GRU\left(x_t, \vec{h}_{t-1}\right) \tag{9}$$

$$h_t = w_t \vec{h}_t + v_t \overleftarrow{h}_t + b_t \tag{10}$$

Here, h_t represents the output state of the forward GRU and the backward GRU at time t, respectively, x_t is the input of the hidden state, W_t and V_t represent the weight matrix of the output state, respectively, b_t represents the bias matrix. Due to the varying contributions of each byte in the packets to the identification of encrypted malicious traffic, we utilize a scaled dot-product attention mechanism to enhance the weight of important features within the temporal feature vectors. Finally, we spliced the temporal features and vectors (h^N_1, ..., $h^N_{[SEP]}$) into a temporal extension vector, avoiding the feature loss of the original byte-level vectors caused by the module series connection.

3.2.3 Spatial Feature Extraction Module

The TextCNN model is a variant model based on Convolutional Neural Networks. We believe that TextCNN is the preferred choice for extracting local multi-receptive field features from traffic, because it exhibits excellent local feature extraction performance in vulnerability text classification [37] and malicious code classification [38] tasks. Compared with traditional CNN, the advantage of TextCNN is that we can set convolution kernels of various sizes to extract spatial features from sequence content

of different lengths. Additionally, the convolutional operations of the TextCNN model can be executed in parallel, significantly accelerating the model training speed. The structure of TextCNN is shown in Fig. 4.

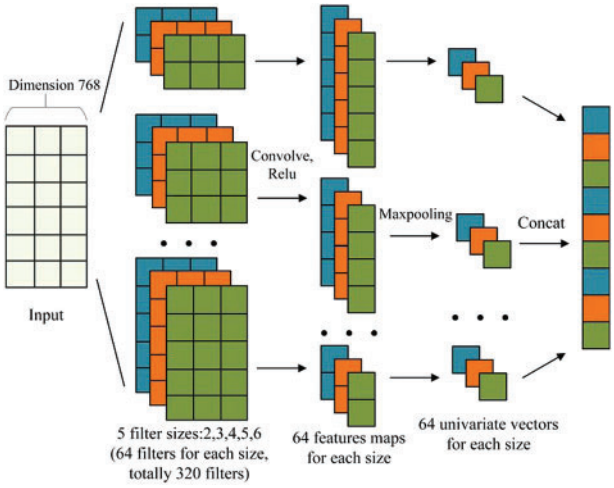


Figure 4: TextCNN model architecture

The convolution kernel of TextCNN can only perform one-dimensional convolution operations in the length direction of the sequence, and its shape is *embedding_size* × *filter_size*. Where *embedding_size* is the dimensionality of the vectors after being embedded by the BERT module, and *filter_size* represents the size (height) h of the convolution kernel, indicating the number of words that can be extracted from the sequence in each convolution operation. The *filter_size* is set to (2, 3, 4,5, 6). The result O_i of each sliding window is calculated through the convolution operation, and its calculation process can be expressed as Eq. (11):

$$o_i = f\left(W_h \cdot x_{i:\, i+h-1} + b_h\right) \tag{11}$$

Here, $x_{i:\, i+h-1}$ is the word vector matrix from row i to row i+h-1, b_h is the bias term, and f represents the nonlinear activation function Relu. Through the height h of the convolution kernel and the sequence length n, it can be calculated that there are n-h+1 convolution windows, so the convolution summary result is c = [O_1, O_2, ..., O_{n-h+1}]. After each filter convolution is completed, a feature map representing the sequence will be extracted. The number of feature maps is equal to the number of convolution kernels.

In the pooling layer, the global maximum pooling method is used to extract the maximum value in each vector. After the pooling operation, we obtain a 64-dimensional vector, which is formed by a combination of the maximum value of the features extracted by each convolution kernel. This not only reduces the feature dimension but also eliminates the length difference between vectors.

Finally, we splice the feature vectors output by convolution kernels of different sizes to obtain local multi-receptive field spatial features. The feature dimension is 5 × *num_filter*, and dropout technology is used to prevent overfitting.

3.3 Encrypted Malicious Traffic Classification Layer

We concatenated the global semantic features and local spatiotemporal features as the final representation of encrypted malicious traffic. Then, we utilized the feature transformation space of

the fully connected layer to map it to the sample category space. Finally, we used the Softmax function to calculate the probabilities for each category of traffic, and selected the maximum probability as the classification result. The Softmax function can be represented by Eq. (12).

$$P_i = \frac{e^{x_i}}{\sum_{i=1}^{k} e^{x_i}} \tag{12}$$

Here, P_i represents the probability of recognizing the traffic as the i class, and x_i is the score for the corresponding category. e is the exponential function with base e, which highlights the larger values in the vector.

4 Experiments

4.1 Experimental Environment and Setup

This experiment was performed using Python version 3.7, with the Windows10 operating system, the processor is an Intel(R) Xeon(R) W-2275 CPU @ 3.30 GHz, and the graphics processing unit is a single NVIDIA RTX A4000, with a 16 GB graphics processing unit memory. All experiments were based on PyTorch and Universal Encoder Representations (UERs) [39].

The model parameters are as follows: In the BERT layer, we set the maximum input length to 128 tokens, and the token embedding dimension to 768. In the BiGRU layer, we set the hidden neurons to 128, the number of attention heads is set to 2, and the number of layers to 2. In the TextCNN convolution layer, we set the number of convolution kernels to 64, chose the global maximum pooling method, selected Relu as the activation function, and set the dropout rate to 0.3. We set the batch size to 64, the learning rate to 2×10^{-5}, and the epoch size to 15, and selected Adam as the model parameter optimizer.

Fig. 5 illustrates the shape of the feature vectors for the main network layers of the BSTFNet model. The model takes 128 tokens as input, which are then transformed into a feature matrix of shape (128, 768) by the embedding layer. The 768-dimensional vector corresponding to the CLS token serves as the global semantic feature of the input traffic sequence, while the representation vectors of the remaining 127 tokens become the input for the local feature extraction layer. A 320-dimensional spatiotemporal feature vector is obtained through the TextCNN layer and the BiGRU layer. Finally, the global semantic features and the local spatiotemporal features are fused to produce the ultimate 1088-dimensional representation vector. Through the fully connected layer and softmax function, traffic classification is achieved.

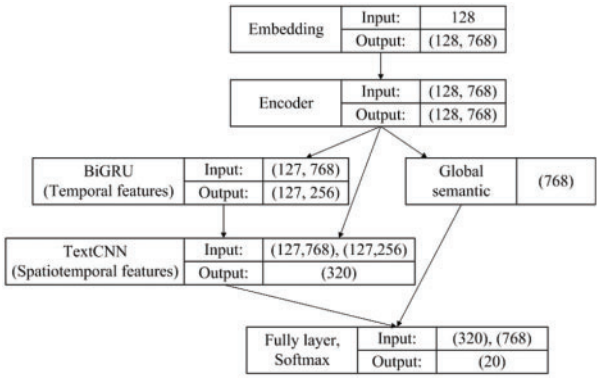


Figure 5: Feature vector shapes for major network layers

4.2 Experimental Datasets

To ensure the credibility of the research results, we chose the USTC-TFC2016 dataset [40] to evaluate the model proposed in this paper. This dataset, resembling a real network environment, has been widely applied in research on encrypted malicious traffic classification. The USTC-TFC2016 dataset is a collection of encrypted traffic, consisting of 10 categories of malware and 10 categories of benign applications. The format of all traffic files is pcap. The dataset contains relatively few redundant and repeated traffic samples, and it is encrypted using the TLS protocol. Table 3 shows the application category composition of the USTC-TFC 2016 dataset.

Table 3: USTC-TFC 2016 dataset composition

Traffic type	Application categories
Benign	BitTorrent, FTP, Warcraft, MySQL, Skype, Facetime, Outlook, SMB, Weibo, Gmail
Malware	Htbot, Cridex, Neris, Nsis-ay, Shifu, Virut, Zeus, Tinba, Miuref, Geodo

4.3 Experimental Evaluation Methods

We evaluated the model's performance using four metrics: Accuracy, Precision, Recall, and F1-score, aiming to assess the model classification performance on encrypted malicious traffic from various perspectives. The specific calculation formulas are represented as Eqs. (13) to (16):

$$\text{Accuracy} = \frac{TP + TN}{FP + TN + FP + TN} \tag{13}$$

$$\text{Precision} = \frac{TP}{FP + TP} \tag{14}$$

$$\text{Recall} = \frac{TP}{FN + TP} \tag{15}$$

$$\text{F1} - \text{score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{16}$$

where TP represents the count of positive samples that have been accurately recognized, while FP corresponds to the count of negative samples that have been erroneously identified. TN signifies the count of negative samples that have been correctly classified, and FN indicates the count of positive samples that have been mistakenly categorized.

4.4 Experimental Results and Analysis

4.4.1 Comparison with Baseline Models

To validate the effectiveness of the model on the encrypted malicious traffic classification task, we selected seven models (FlowPrint [41], FS-Net [24], 1D-CNN, TextCNN, CNN-LSTM [26], ET-BERT [4], BFCN [31]) as baseline models. The experiments were conducted on the USTC-TFC2016 dataset to demonstrate the generalization capability of the BSTFNet model. Table 4 presents the experimental results of the BSTFNet model and other baseline models on the USTC-TFC2016 dataset.

Table 4: Comparison results on malicious traffic classification

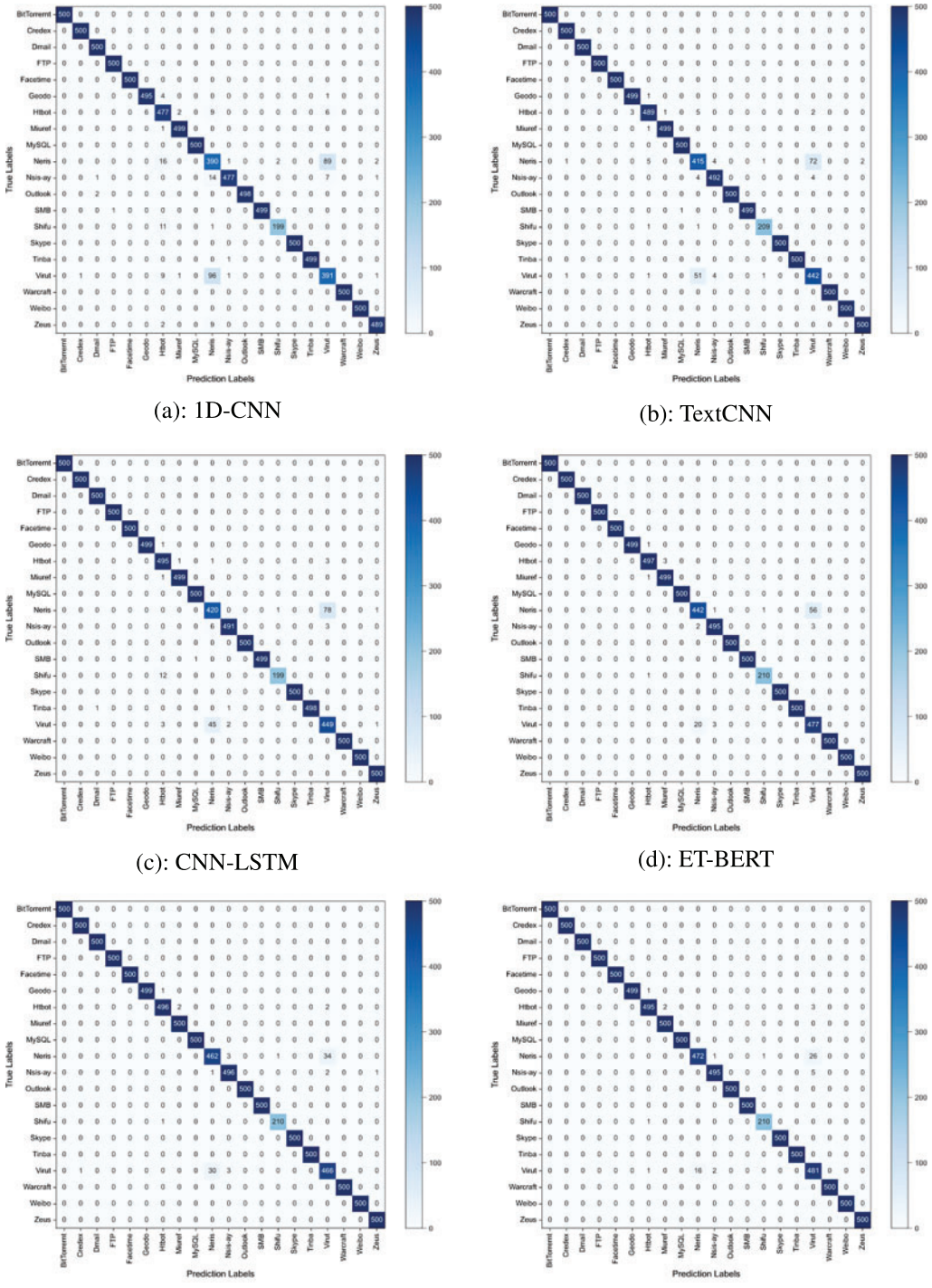
Method	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
FlowPrint	81.46	65.34	70.02	65.73
FS-Net	88.46	88.46	89.20	88.40
1D-CNN	96.93	97.04	96.86	96.94
TextCNN	98.28	98.30	98.30	98.30
CNN-LSTM	98.32	98.37	98.21	98.27
ET-BERT	99.05	99.09	99.07	99.07
BFCN	99.16	99.17	99.17	99.17
Proposed	99.39	99.40	99.40	99.40

The experimental results indicate that our approach achieved the best performance across all four evaluation metrics and outperformed all baseline models. The accuracy, precision, recall, and F1-score of the BSTFNet model are 99.39%, 99.40%, 99.40%, and 99.40%, respectively. This represents an improvement of 2.46%, 2.36%, 2.54%, and 2.46% over the 1D-CNN model, and a 0.34%, 0.31%, 0.33%, and 0.33% improvement over the ET-BERT model. What is impressive is that compared with the BFCN model composed of BERT and CNN, our model improved by 0.23% in four indicators, indicating that the BSTFNet model can make up for the shortcomings of the BFCN model in extracting temporal features and spatial features.

Fig. 6 displays the confusion matrices for well-performing models (1D-CNN, TextCNN, CNN-LSTM, ET-BERT, BFCN, BSTFNet) on the USTC-TFC2016 dataset. Confusion matrices are commonly used to validate classification results and enable researchers to gain a better understanding of the model performance. The rows in the confusion matrix correspond to the actual categories of the samples, and the columns represent the predicted labels inferred by the models. It can be observed from Fig. 6 that most models exhibit some confusion between the Neris and Virut categories. In the case of our model, samples predicted as Neris class included 16 Virut samples, 2 Nsis-ay samples, and 1 Htbot sample. Samples predicted as Virut included 26 Neris samples, 5 Nsis-ay samples, and 3 Htbot samples. Compared to the ET-BERT model, our model reduced the confusion of samples by 30 in the Neris class and by 4 in the Virut class. For a more intuitive comparison, Table 5 statistics the Precision, Recall, and F1-score of the six models in the Neris and Virut categories. The table shows that the BSTFNet model performs the best in the Neris and Virut categories, with respective improvements of 3.85% and 1.93% in F1-score compared to the baseline model ET-BERT. This indicates that the BSTFNet model exhibits superior detection performance for malicious traffic types with higher similarities.

4.4.2 Model Parameter Analysis

Fig. 7 shows the training curve of the BSTFNet model. We can observe the convergence behavior of the model on the data set. The BSTFNet model achieved convergence after only 10 training epochs. After reaching convergence, the accuracy of the model on the training set and validation set reached 100% and 99.40%, respectively. The experimental results confirm the robust performance of the model on the data set, showing excellent convergence speed and accuracy, and no signs of overfitting.



(a): 1D-CNN

(b): TextCNN

(c): CNN-LSTM

(d): ET-BERT

(e): BFCN

(f): BSTFNet

Figure 6: Confusion matrices for six models

Table 5 : Experimental results of six models in the nerius class and Virut class

Method	Nerius class			Virut class		
	Precision (%)	Recall (%)	F1 (%)	Precision (%)	Recall (%)	F1 (%)
1D-CNN	80.70	73.60	77.00	77.21	84.00	80.46
TextCNN	87.18	83.00	85.04	85.00	88.40	86.67
CNN-LSTM	88.98	84.00	86.42	84.30	90.20	87.15
ET-BERT	95.26	88.40	91.70	88.99	95.40	92.85
BFCN	93.71	92.40	93.51	92.46	93.20	92.83
BSTFNet	96.72	94.40	95.55	93.40	96.20	94.78

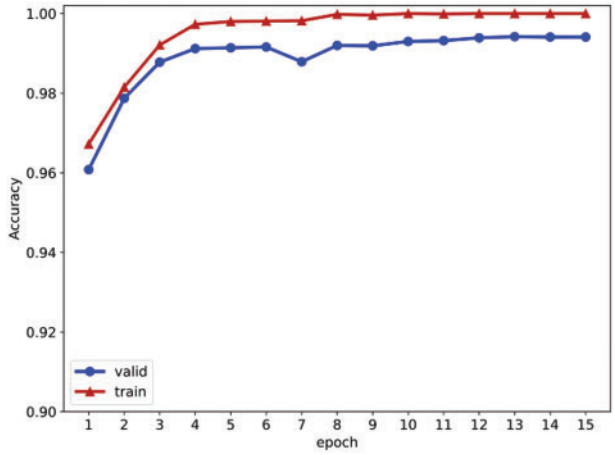


Figure 7: Training curve of BSTFNet model

The convolution kernel size is the core parameter of the TextCNN model. The local spatial feature extraction layer aims to extract local multi-scale spatial features by defining convolution kernels of different sizes, thereby obtaining diverse and more representative features. We selected 5 commonly used convolution kernel size combinations for the experiments. The experimental results are shown in Fig. 8. The results show that the model using convolution kernel sizes of 2, 3, 4, 5, and 6 has the highest accuracy on the data set, reaching 99.39%.

Num_layer represents the number of stacked layers of bidirectional GRU, which is closely related to the complexity and fitting ability of the BiGRU module. Each layer of GRU can capture sequence features at different time steps. If *Num_layer* is too small, the model is insufficient to extract the deep features of traffic. Conversely, the model will overfit the fine information of the samples, resulting in a decrease in generalization ability. Moreover, as the GRU layers are stacked too much, the memory cost of the model will also increase exponentially. Therefore, by choosing the appropriate number of layers, the model can have both excellent generalization and fitting capabilities. To select parameters suitable for the model, we designed four stacking layer numbers for experiments. The experimental results are shown in Fig. 9. When *Num_layer* is 2, the model has the highest accuracy in the encrypted malicious traffic classification task. At the same time, the results verify that too many or too few GRU layers will bring the risk of over-fitting and under-fitting to the model.

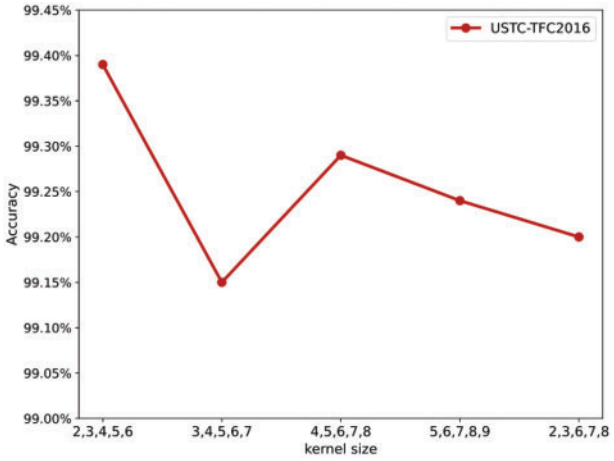


Figure 8: Effect of different convolution kernel sizes

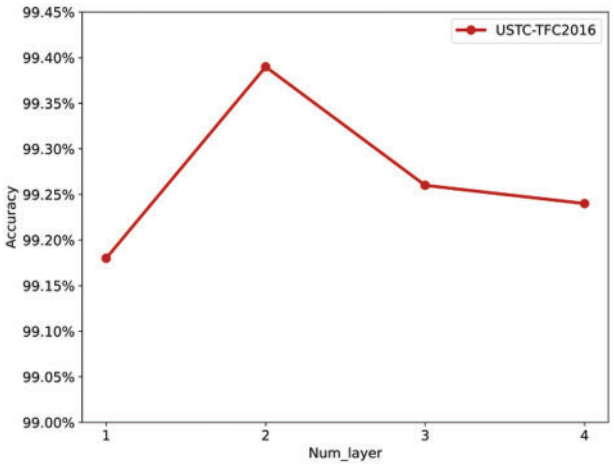


Figure 9: Effect of different *num_layer*

4.4.3 Ablation Experiments

To validate the indispensability of each feature extraction module in our model, we designed four ablation models for experimental comparison: BERT-TextCNN, BERT-BiGRU, BERT-NOCLS, and TextCNN-BiGRU. The BERT-TextCNN model removes the BiGRU-Attention temporal feature extraction module, BERT-BiGRU removes the TextCNN local spatial feature extraction module, TextCNN-NOCLS removes the global semantic features, and TextCNN-BiGRU removes the BERT encoder. All experiments are conducted on the processed USTC-TFC2016 data set. Fig. 10 is the harmonic mean of all category results in the data set. Figs. 11 and 12 are the experimental results for the Nerius category and Virut category, respectively.

The experimental results show that compared with the other four models, the BSTFNet model achieved the best performance in the four average indicators of all categories, increasing the F1-score by 0.86%, 0.22%, 0.19%, and 0.17%, respectively. For the nerius class, which is more difficult to classify, the F1-score increased by 8.37%, 2.76%, 2.43%, and 2.05%, respectively, and for the virut class, they increased by 7.80%, 1.75%, 1.39%, and 0.99%, respectively. After removing the BERT layer,

the model performance dropped sharply, indicating that the BERT module of the BSTFNet model is crucial for mining high-dimensional features. After removing the temporal feature extraction module, spatial feature extraction module, and global semantic features respectively, the model performance also declined, indicating that the local spatiotemporal features extracted by the BiGRU and TextCNN models can greatly improve the attention and identification ability of encrypted malicious traffic. Models with global semantic features are better able to extract the deep contextual relationships of encrypted malicious traffic, and the parallel connection of global features and local features avoids the loss of some features caused by series connection to a certain extent. Through the above experiments, we verified the necessity of each feature extraction module of the BSTFNet model, indicating that a model that simultaneously considers global features, temporal features, and spatial features performs better than a model that only considers a single global semantic feature.

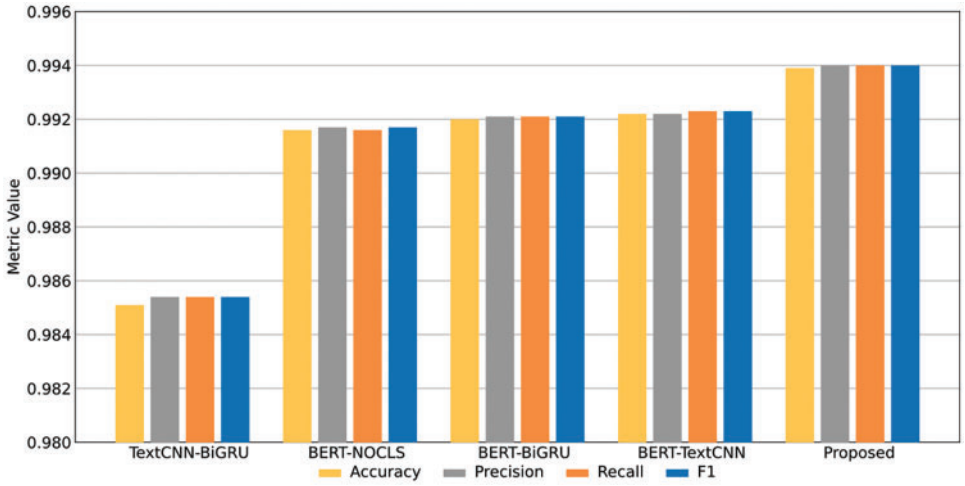


Figure 10: Average accuracy, recall, precision, and F1-score for each ablation model across all categories

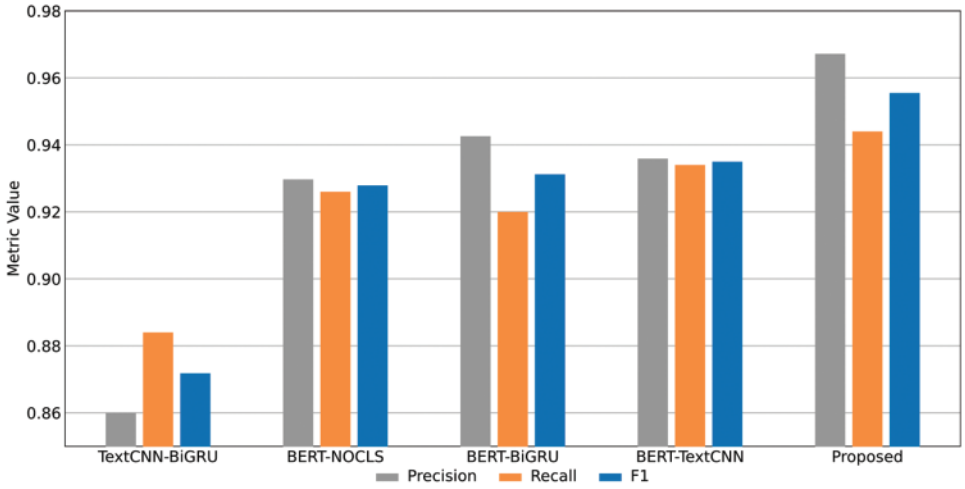


Figure 11: The recall rate, precision rate, and F1-score of each ablation model on the Nerius category

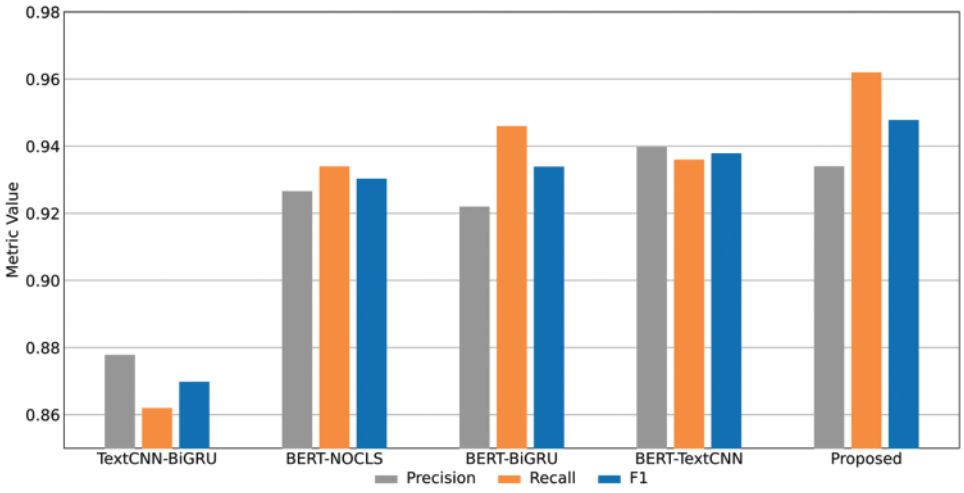


Figure 12: The recall rate, precision rate, and F1-score of each ablation model on the Virut category

5 Conclusion

Aiming at the problems of existing encrypted malicious traffic classification methods with single representation and insufficient feature extraction depth, this paper proposes an encrypted malicious traffic classification method based on global semantic features and local spatiotemporal features. We first use BERT to extract the global semantics of traffic, and then use TextCNN and BiGRU models to extract the spatial and temporal characteristics of traffic respectively. The experimental results show that our model achieves the optimal accuracy and F1-score on the USTC-TFC2016 data set. At the same time, the ablation experiment proves the indispensability of each module of the BSTFNet model. In future work, we will start from the data level and use data enhancement technology to deal with the problem of unbalanced encryption malicious traffic classification to further adapt to the actual network environment. On the other hand, the detection of malicious traffic requires high immediacy. In the future, we will combine real-time machine learning and deep learning to achieve fine-grained classification of encrypted malicious traffic under the big data platform, and support real-time updates of the model.

Acknowledgement: The authors would like to thank the reviewers for their contribution to this paper.

Funding Statement: This research was funded by National Natural Science Foundation of China under Grant No. 61806171, Sichuan University of Science & Engineering Talent Project under Grant No. 2021RC15, Open Fund Project of Key Laboratory for Non-Destructive Testing and Engineering Computer of Sichuan Province Universities on Bridge Inspection and Engineering under Grant No. 2022QYJ06, Sichuan University of Science & Engineering Graduate Student Innovation Fund under Grant No. Y2023115, The Scientific Research and Innovation Team Program of Sichuan University of Science and Technology under Grant No. SUSE652A006.

Author Contributions: Study conception and design: Hong Huang, Xingxing Zhang; data collection: Xingxing Zhang, Ye Lu, Ze Li; analysis and interpretation of results: Hong Huang, Xingxing Zhang, Shaohua Zhou; draft manuscript preparation: Xingxing Zhang. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Publicly available datasets were analyzed in this study. This data can be found here: https://github.com/yungshenglu/USTC-TFC2016.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] K. Keshkeh, A. Jantan, K. Alieyan, and U. M. Gana, "A review on TLS encryption malware detection: TLS features, machine learning usage, and future directions," in *Proc. Adv. Cyber Secur.*, Penang, Malaysia, 2021, pp. 213–229.

[2] H. Tahaei, F. Afifi, A. Asemi, F. Zaki, and N. B. Anuar, "The rise of traffic classification in IoT networks: A survey," *J. Netw. Comput. Appl.*, vol. 154, pp. 102538, 2020. doi: 10.1016/j.jnca.2020.102538.

[3] E. Nazarenko, V. Varkentin, and T. Polyakova, "Features of application of machine learning methods for classification of network traffic (features, advantages, disadvantages)," in *Proc. 2019 Int. Multi-Conf. Indust. Eng. Mod. Technol. (FarEastCon)*, Vladivostok, Russia, 2019, pp. 1–5.

[4] X. Lin *et al.*, "ET-BERT: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proc. ACM Web Conf. 2022*, Lyon, France, 2022, pp. 633–642.

[5] E. Papadogiannaki and S. Ioannidis, "A survey on encrypted network traffic analysis applications, techniques, and countermeasures," *ACM Comput. Surv.*, vol. 54, no. 123, pp. 1–35, 2021. doi: 10.1145/3457904.

[6] J. McPherson, K. L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen, "PortVis: A tool for port-based detection of security events," in *Proc. 2004 ACM Workshop Vis. Data Min. Comput. Secur.*, Washington, USA, 2004, pp. 73–81.

[7] A. Madhukar and C. Williamson, "A longitudinal study of P2P traffic classification," in *Proc. 14th IEEE Int. Symp. Model., Anal. Simul.*, Monterey, CA, USA, 2006, pp. 179–188.

[8] C. Thay, V. Visoottiviseth, and S. Mongkolluksamee, "P2P traffic classification for residential network," in *Proc. 2015 Int. Comput. Sci. Eng. Conf. (ICSEC)*, Chiang Mai, Thailand, 2015, pp. 1–6.

[9] M. Korczyński and A. Duda, "Markov chain fingerprinting to classify encrypted traffic," in *Proc. IEEE INFOCOM 2014-IEEE Conf. Comput. Commun.*, Toronto, ON, Canada, 2014, pp. 781–789.

[10] C. Li, Y. Guo, and X. Wang, "Towards privacy-preserving dynamic deep packet inspection over outsourced middleboxes," *High-Confid. Comput.*, vol. 2, pp. 100033, 2022. doi: 10.1016/j.hcc.2021.100033.

[11] J. Ning, G. S. Poh, J. C. N. Loh, J. Chia, and E. C. Chang, "PrivDPI: Privacy-preserving encrypted traffic inspection with reusable obfuscated rules," in *Proc. 2019 ACM SIGSAC Conf. Comput. Commun. Secur.*, London, UK, 2019, pp. 1657–1670.

[12] A. S. Iliyasu and H. Deng, "Semi-supervised encrypted traffic classification with deep convolutional generative adversarial networks," *IEEE Access*, vol. 8, pp. 118–126, 2020. doi: 10.1109/ACCESS.2019.2962106.

[13] F. M. Alotaibi and F. M. Alotaibi, "Network intrusion detection model using fused machine learning technique," *Comput. Mater. Contin.*, vol. 75, no. 2, pp. 2479–2490, 2023.

[14] B. Anderson, S. Paul, and D. McGrew, "Deciphering malware's use of TLS (without decryption)," *J. Comput. Virol. Hacking Tech.*, vol. 14, pp. 195–211, 2018.

[15] M. J. D. Lucia and C. Cotton, "Detection of encrypted malicious network traffic using machine learning," in *Proc. MILCOM 2019-2019 IEEE Military Commun. Conf. (MILCOM)*, Norfolk, VA, USA, 2019, pp. 1–6.

[16] A. S. Shekhawat, F. D. Troia, and M. Stamp, "Feature analysis of encrypted malicious traffic," *Expert. Syst. Appl.*, vol. 125, pp. 130–141, 2019. doi: 10.1016/j.eswa.2019.01.064.

[17] I. Torroledo, L. D. Camacho, and A. C. Bahnsen, "Hunting malicious TLS certificates with deep neural networks," in *Proc. 11th ACM Workshop Artif. Intell. Secur.*, Toronto, Canada, 2018, pp. 64–73.

[18] T. Shapira and Y. Shavitt, "FlowPic: Encrypted internet traffic classification is as easy as image recognition," in *Proc. IEEE INFOCOM 2019-IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Paris, France, 2019, pp. 680–687.

[19] F. Jie, "Research on malicious TLS traffic identification based on hybrid neural network," in *Proc. 2020 Int. Conf. Adv. Ambient Comput. Intell. (ICAACI)*, Ottawa, ON, Canada, 2020, pp. 42–46.
[20] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. 2017 IEEE Int. Conf. Intell. Secur. Inform. (ISI)*, Beijing, China, 2017, pp. 43–48.
[21] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Comput.*, vol. 24, pp. 1999–2012, 2020. doi: 10.1007/s00500-019-04030-2.
[22] R. H. Hwang, M. C. Peng, C. W. Huang, P. C. Lin, and V. L. Nguyen, "An unsupervised deep learning model for early network traffic anomaly detection," *IEEE Access*, vol. 8, pp. 30387–30399, 2020. doi: 10.1109/ACCESS.2020.2973023.
[23] M. Chen, X. Wang, M. He, L. Jin, K. Javeed and X. Wang, "A network traffic classification model based on metric learning," *Comput. Mater. Contin.*, vol. 64, no. 2, pp. 941–959, 2020. doi: 10.32604/cmc.2020.09802.
[24] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: A flow sequence network for encrypted traffic classification," in *Proc. IEEE INFOCOM 2019-IEEE Conf. Comput. Commun.*, Paris, France, 2019, pp. 1171–1179.
[25] K. Lin, X. Xu, and H. Gao, "TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of IIoT," *Computer. Netw.*, vol. 190, pp. 107974, 2021. doi: 10.1016/j.comnet.2021.107974.
[26] Z. Zou *et al.*, "Encrypted traffic classification with a convolutional long short-term memory neural network," in *Proc. 2018 IEEE 20th Int. Conf. High Perform. Comput. Commun.; IEEE 16th Int. Conf. Smart City; IEEE 4th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Exeter, UK, 2018, pp. 329–334.
[27] X. Liu *et al.*, "Attention-based bidirectional GRU networks for efficient HTTPS traffic classification," *Inform. Sci.*, vol. 541, pp. 297–315, 2020. doi: 10.1016/j.ins.2020.05.035.
[28] K. Wang, J. Gao, and X. Lei, "MTC: A multi-task model for encrypted network traffic classification based on transformer and 1D-CNN," *Intell. Autom & Soft Comput.*, vol. 37, no. 1, pp. 619–638, 2023. doi: 10.32604/iasc.2023.036701.
[29] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," arXiv:1810.04805, 2018.
[30] H. Y. He, Z. G. Yang, and X. N. Chen, "PERT: Payload encoding representation from transformer for encrypted traffic classification," in *Proc. 2020 ITU Kaleidoscope: Ind.-Driv. Digital Transform. (ITU K)*, Ha Noi, Vietnam, 2020, pp. 1–8.
[31] Z. Shi, N. Luktarhan, Y. Song, and G. Tian, "BFCN: A novel classification method of encrypted traffic based on BERT and CNN," *Electron.*, vol. 12, no. 3, pp. 516, 2023. doi: 10.3390/electronics12030516.
[32] X. Meng, C. C. Lin, Y. Wang, and Y. Zhang, "NetGPT: Generative pretrained transformer for network traffic," arXiv:2304.09513, 2023.
[33] P. Wang, S. Li, F. Ye, Z. Wang, and M. Zhang, "PacketCGAN: Exploratory study of class imbalance for encrypted traffic classification using CGAN," in *Proc. ICC 2020-2020 IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, 2020, pp. 1–7.
[34] M. Wankhade and A. C. S. Rao, "Opinion analysis and aspect understanding during COVID-19 pandemic using BERT-Bi-LSTM ensemble method," *Sci. Rep.*, vol. 12, pp. 17095, 2022. doi: 10.1038/s41598-022-21604-7.
[35] R. Pandey and J. P. Singh, "BERT-LSTM model for sarcasm detection in code-mixed social media post," *J. Intell. Inf. Syst.*, vol. 60, pp. 235–254, 2023. doi: 10.1007/s10844-022-00755-z.
[36] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light gated recurrent units for speech recognition," *IEEE Trans. on Emerg. Topics Comput. Intell.*, vol. 2, pp. 92–102, 2018. doi: 10.1109/TETCI.2017.2762739.
[37] M. Pan, P. Wu, Y. Zou, C. Ruan, and T. Zhang, "An automatic vulnerability classification framework based on BiGRU-TextCNN," *Procedia Comput. Sci.*, vol. 222, pp. 377–386, 2023. doi: 10.1016/j.procs.2023.08.176.

[38] Q. Wang and Q. Qian, "Malicious code classification based on opcode sequences and textCNN network," *J. Inform. Secur. Appl.*, vol. 67, pp. 103151, 2022. doi: 10.1016/j.jisa.2022.103151.
[39] Z. Zhao *et al.*, "UER: An open-source toolkit for pre-training models," arXiv:1909.05658, 2019.
[40] W. Wang, M. Zhu, X. W. Zeng, X. Z. Ye, and Y. Q. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. 2017 Int. Conf. Inform. Netw. (ICOIN)*, Da Nang, Vietnam, 2017, pp. 712–717.
[41] T. van Ede *et al.*, "FlowPrint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," in *Proc. Netw Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, California, USA, 2020.