electronics

MDPI

Article

BFCN: A Novel Classification Method of Encrypted Traffic Based on BERT and CNN

Zhaolei Shi [1], Nurbol Luktarhan [1,*], Yangyang Song [1] and Gaoqi Tian [2]

[1] College of Information Science and Engineering, Xinjiang University, Urumqi 830046, China
[2] School of Software, Xinjiang University, Urumqi 830046, China
* Correspondence: nurbol@xju.edu.cn

Abstract: With the speedy advancement of encryption technology and the exponential increase in applications, network traffic classification has become an increasingly important research topic. Existing methods for classifying encrypted traffic have certain limitations. For example, traditional approaches such as machine learning rely heavily on feature engineering, deep learning approaches are susceptible to the amount and distribution of labeled data, and pretrained models focus merely on the global traffic features while ignoring local features. To solve the above problem, we propose a BERT-based byte-level feature convolutional network (BFCN) model consisting of two novel modules. The first is a packet encoder module, in which we use the BERT pretrained encrypted traffic classification model to capture global traffic features through its attention mechanism; the second is a CNN module, which captures byte-level local features in the traffic through convolutional operations. The packet-level and byte-level features are concatenated as the traffic's final representation, which can better represent encrypted traffic. Our approach achieves state-of-the-art performance on the publicly available ISCX-VPN dataset for the traffic service and application identification task, achieving F1 scores of 99.11% and 99.41%, respectively, on these two tasks. The experimental results demonstrate that our method further improves the performance of encrypted traffic classification.

Keywords: encrypted traffic classification; convolutional neural network; pretraining; bidirectional encoder representations from transformers

Citation: Shi, Z.; Luktarhan, N.; Song, Y.; Tian, G. BFCN: A Novel Classification Method of Encrypted Traffic Based on BERT and CNN. *Electronics* **2023**, *12*, 516. https://doi.org/10.3390/electronics12030516

Academic Editor: Wojciech Mazurczyk

Received: 23 December 2022
Revised: 7 January 2023
Accepted: 16 January 2023
Published: 19 January 2023

1. Introduction

Network traffic classification refers to classifying network traffic into particular application classes, an essential technique in network security and management [1]. It performs a considerably important role in tasks related to intrusion detection, traffic trend analysis, quality of service (QoS), quality of experience (QoE), and network visibility [2]. In recent decades, traffic encryption technologies have significantly advanced. Internet applications and websites have adopted encryption protocols for their data due to the growing concern of users and enterprises for privacy and confidentiality. Traffic using encryption protocols, such as Secure Sockets Layer (SSL)/Transport Layer Security (TLS), not only provides Internet users with freedom, privacy, and anonymity but also poses tremendous challenges and obstacles to classifying encrypted traffic. Malicious software and cybercriminals can exploit the privacy and anonymity offered by encryption technology to circumvent firewall detection and monitoring systems [3], jeopardizing the interests of Internet users, which brings enormous potential risks and issues to network security and network resource management. Traffic encryption protocols make it extremely difficult to identify and classify malicious or exfiltration traffic that can endanger users; therefore, encrypted traffic classification has become a hot topic in network security and has raised interest among a large number of researchers in both academia and industry [4].

With the fast advancement and widespread adoption of encryption techniques, some traditional methods cannot be applied to encrypted traffic since they function based on

plaintext packet loads, such as deep packet inspection (DPI) [5]. Previous work utilized the leftover plain text of encrypted traffic (e.g., size, certificates) to construct fingerprints and then matched the fingerprints against the traffic [6,7]. However, this method does not work for emerging encryption techniques (e.g., TLS 1.3), as the plain text in traffic using this encryption technique becomes sparse or ambiguous, making it challenging to construct fingerprints. Subsequent research turned to using the statistical features of encrypted traffic to classify traffic and using classical machine learning algorithms that performed well in various fields to identify that encrypted traffic without plaintexts. However, the performance of this kind of approach is heavily dependent on the quality of the expert-designed features, has limited generalization capability, and is very prone to human-generated mistakes [8]. Along with the remarkable success of deep learning in text classification, image classification, and sentiment recognition, the field of encrypted traffic classification has also been extensively researched using deep learning and has achieved significant performance improvements. Deep learning has the advantage of extracting complex features automatically [9]. However, models using deep learning are susceptible to the amount and distribution of labeled data, resulting in models learning biased features.

A significant breakthrough in natural language processing occurred when Google's AI language team publicly opened the source of a pretrained model known as the bidirectional encoding representation transformer (BERT), which has become popular in academia and industry because of its state-of-the-art results in a number of different domains. The pretrained model can acquire unbiased general data features from a massive volume of unlabeled data; the model can fine-tune a small number of labeled data to obtain excellent performance [10]. Recently, BERT has also been introduced in encrypted traffic classification and achieved state-of-the-art performance on several datasets. However, this approach ignores byte-level features in the traffic and predicts the class of encrypted traffic by packet-level features only [8]. In the field of encrypted traffic classification, although existing methods achieve good results, there is still the problem of needing to adequately exploit the features of data packets.

In this paper, we propose a novel model, namely the byte feature convolutional network (BFCN), to improve the performance of encrypted traffic classification. We implemented the following strategies to address the shortcomings of previous pretrained models in encrypted traffic classification. We used the BERT model to extract packet-level feature vector representations of traffic and then captured byte-level locally salient feature vectors using convolutional operations. These two feature vectors concatenate as the ultimate traffic feature representation. The major contribution of this paper is to propose a BFCN model based on BERT and convolutional neural networks (CNNs) for the problem of encrypted traffic classification. The BFCN model can enhance the performance of encrypted traffic classification by capturing byte-level and packet-level features and concatenating them as the final traffic features. That traffic's final representation can better represent encrypted traffic. Our work leverages the traffic features more than previous work. Our proposed method reached new state-of-the-art F1 values of 99.11% and 99.41% for the service identification and application identification tasks on the ISCX-VPN dataset, respectively. The experimental results demonstrate that our idea is correct and effective.

2. Related Work

2.1. Encrypted Traffic Classification

Network traffic classification tasks can be grouped into the following three types:

- Encrypted traffic identification task [11], whose purpose is to classify traffic into encrypted and non-encrypted traffic;
- Service identification task, whose aim is to identify the service type of various applications to which the traffic belongs, such as chat applications, file transfer applications, etc.;
- Traffic application identification task [9,12], whose aim is to identify the concrete applications to which the traffic belongs, such as Youtube, Gmail, etc.;

In this section, we summarize the most critical network traffic classification methods, Table 1 shows these methods. According to the techniques used, the network traffic classification approaches are divided into the following four categories: port-based, constructive fingerprinting, statistical methods, and deep learning models.

Port-based: Port-based network traffic classification methods are the most ancient and well known [13]. Since each application is assigned a specific port number by the Internet Assigned Numbers Authority (IANA), e.g., port 21 for applications using the FTP protocol and port 22 for applications using the SSH protocol, the port-based method of classifying network traffic can classify network traffic using the port information in the transport layer. Because of the short time consumption, low cost-effectiveness, and high accuracy of port-based methods, firewall technology and access control lists (ACLs) frequently utilize this approach [14]. Although port-based methods are not affected by encryption protocols, with the advancement of Internet technologies, several techniques and methods have emerged that drastically decrease the accuracy of port-based methods, such as random port policies [15], port masquerading techniques, network address translation (NAT) protocol, etc. Hence, in the contemporary network environment, it is necessary to identify the current traffic using higher-level methods. Notably, our model does not rely on any port number.

Fingerprint construction: There are two major classes of methods based on constructive fingerprinting techniques, one represented by deep packet inspection (DPI) [9] and the other by FlowPrint [6]. DPI inspects the overall packet content, including the header and payload. If some predefined fixed string patterns are detected anywhere in the packet, it can determine the traffic class, and some studies in the literature refer to these patterns as the signature [11]. The evolution and widespread use of encryption protocols has contributed to the payload being in a pseudo-random format, so DPI does not apply to encrypted traffic tasks. FlowPrint [6] uses unencrypted protocol field information, such as size, certificate, device, and time characteristics, to represent each flow. This approach is highly accurate and can identify traffic earlier based on the first few packets of the network flow. However, these methods are highly dependent on plaintext information, and such information can be prone to be tampered with during transmission, thus losing the correct meaning [8], while our model does not need to depend on plaintext information within the packet.

Statistical methods: There are differences in the traffic characteristics of various application classes, and researchers can exploit these differences to identify different traffic flows using machine learning models. AppScanner [16] uses the statistical features of packet size to train random-forest classifiers, while BIND [17] uses time-dependent statistical features. Statistical feature-based methods do not have to inspect the packet contents by byte, so the computational complexity is much lower. Since statistical approaches utilize only feature data and do not require detecting the packet content, they are able to identify the encrypted traffic. However, these methods have significant drawbacks, as feature extraction and feature selection require the support of specialists, making these methods time-consuming, costly, laborious, and prone to human mistakes [8]. By contrast, our model does not depend on human-designed features.

Deep learning models: Since deep learning models do not depend on hand-designed features, they can automatically capture features from raw traffic and achieve high accuracy rates; thus, the use of deep learning models to identify encrypted traffic is favored by academia and industry. DF [18] uses convolutional neural networks to automatically extract representations of traffic from raw packet size sequences of encrypted traffic, while FS-NET [19] uses recurrent neural networks (RNNs). DeepPacket [9] and TSCRNN [20] pack feature extraction and classifiers into a framework that, when the payload in the original packet is fed into the model, automatically completes the feature extraction and classification operations. However, using deep learning methods that depend on a large volume of labeled data, the method's performance is highly vulnerable to the imbalance of the dataset [8]. In comparison, our model does not need to depend on a large number of labeled datasets.

2.2. Pretraining Models

Recent research has shown that pretrained models can learn generic language representations on large corpora, which is beneficial for downstream tasks, avoiding the necessity to train models from the beginning [21]. It is sufficient to improve the pretrained model for a given task and then fine-tune it. The most widely influenced is the bidirectional encoder representations from the transformer model published by Google [10], which employs an unsupervised masked language model (MLM) for training that randomly masks some tokens for prediction.

In encrypted traffic, despite the absence of explicit semantics of the payload in the packets, Sengupta [22] exploited randomness differences between different encrypted traffic to identify the traffic, demonstrating that the encrypted traffic is not absolute and perfectly random, but there exist implicit patterns. Computer communication protocols and natural languages have some common features [23], so the BERT can be transformed to perform well in crypto traffic variety. PERT [23] firstly implemented a pretraining model for encrypted traffic classification; it migrated ALBERT to perform the service identification task on the ISCX-VPN dataset and obtained an F1 of 93.23%, but the PERT model is not designed to characterize the traffic representation and pretraining tasks corresponding to it, which leads to its poor performance and limits the generalization ability of the model. CDB [24] comprises a one-dimensional CNN and a transformer's encoder; the method pretrains on unlabeled data to learn the basic features of encrypted traffic and then transfers to other datasets to complete the classification of encrypted traffic. ET-BERT [8] was designed with two pretraining tasks. The model learned packet-level representations by pretraining on a large volume of unlabeled data and then fine-tuning with a small number of task-specific labeled data. The method achieved state-of-the-art performance on five encrypted traffic classification tasks. The F1 values reached 98.90% and 99.37% for the service and application identification tasks performed on the ISCX-VPN dataset. However, we observe that the model merely focuses on packet-level features and lacks consideration of the byte-level features of encrypted traffic.

Table 1. Related work in encrypted traffic classification.

Method	Paper
Port-based	N/A
Fingerprint construction	Deep Packet Inspection (DPI) [9], FlowPrint [6]
Statistical methods	AppScanner [16], BIND [17]
Deep learning model	DF [18], FS-NET [19], DeepPacket [9], TSCRNN [20]
Pretraining models	PERT [23], CDB [24], ET-BERT [8]

3. Methodology

3.1. Model Architecture

In recent years, the combination of both BERT and CNN has been successfully implemented in the field of natural language processing (NLP), such as sentiment classification [25] and text classification [21], and has obtained excellent performance. Inspired by these research works, we proposed a byte feature convolutional (BFCN) model, whose network structure is illustrated in Figure 1. Our approach mainly consists of a packet encoder layer and a byte feature convolutional layer, which employ a BERT module and a convolution module, respectively. In the packet encoder layer, we not only transformed the encrypted traffic into the input vector of the byte feature convolutional layer but also used the state in the ultimate layer corresponding to token [CLS] as the packet-level features for the later classification task. Compared with other tokens, the token [CLS] with no obvious information can incorporate the semantic information of each token in the packet more fairly under the attention mechanism. Thus, it can better represent the features of the whole packet. We utilized the convolution operation in the byte feature convolutional layer to extract locally salient features. Both features were concatenated as the ultimate

traffic feature representation. Finally, we used a fully connected layer as our classifier for predicting the class to which the traffic belongs.

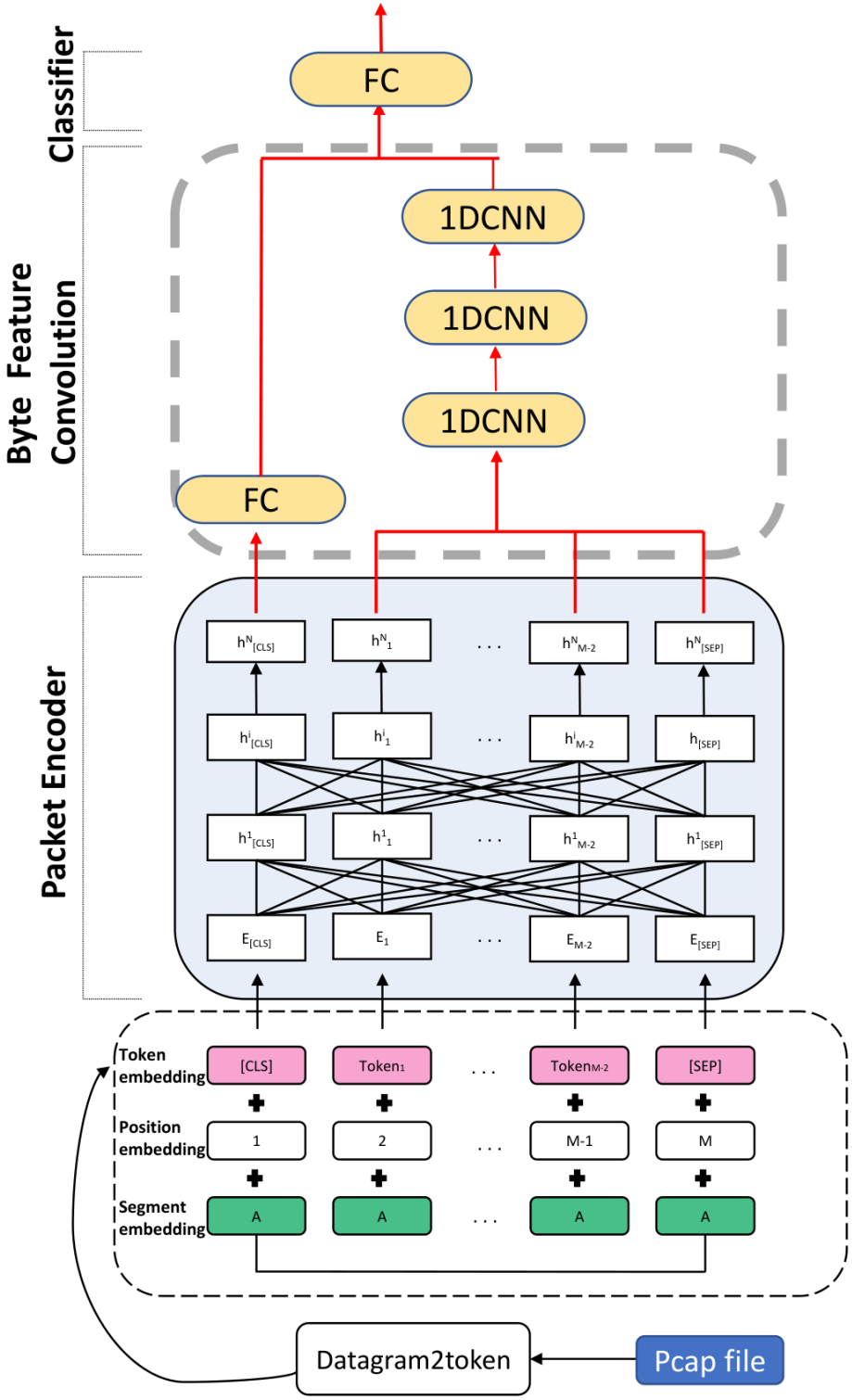

Figure 1. Byte feature convolution model network (BFCN) structure.

Compared with previous studies in [21,25], the novelty of our model is as follows: In [25], the states of the final hidden layers of all tokens are created as global features. In contrast, our approach uses the hidden layer states associated with the [CLS] token as global features. Compared with other tokens, the [CLS] token has no specific meaning

and can fairly fuse each token's information under the attention mechanism. The authors of [21] input the final hidden layer states of all tokens except the [CLS] token into a module consisting of pooling and convolutional layers to learn the local features, while in our approach, we input the final hidden layer states of all tokens except the [CLS] token into a module consisting of only 1D-CNN to learn the local features.

The form of input to the model is the sum of token embedding, position embedding, and segmentation embedding.

Token embedding: We read the preprocessed raw traffic data and then used byte-pair encoding for token representation, where each token unit ranges from 0 to 65,535, and the dictionary size $|V|$ is max expressed as 65,536; the representation of the token learned from the lookup table is called token embedding [8]. In addition, each sequence starts with a token [CLS], and the state of the final hidden layer associated with this token represents the packet-level feature vector. Compared with other tokens, the token [CLS] with no obvious information can incorporate the semantic information of each token in the packet more fairly under the attention mechanism. Thus, it can better represent the features of the whole packet. Token [PAD] is a padding notation used when the data do not meet the minimum length requirement. Token [SEP] is added to the last of each sequence addition.

Position embedding: Since traffic data are transmitted in order, the traffic data are time-series in nature. We used positional embedding to ensure that the model can recognize the temporal relationship between different tokens [8].

Segment embedding: In the pretraining phase, two segments may be input at once; this is not the case in our work, where each packet represents one segment.

3.2. Datasets

To validate our method, for our experiments, we selected the ISCX VPN-nonVPN traffic dataset (ISCX-VPN) [26], published by the University of New Brunswick, which includes the raw traffic data generated at different applications. Based on the actions conducted by the application at the moment of capturing the traffic (e.g., chats, file transfers, or video calls, etc.) and the application from which the traffic originates (e.g., Youtube, Facebook, Netflix, etc.), the traffic was labeled with a corresponding label. In our experiments, we performed service and application identification tasks on this dataset. Service categories included 12 categories, and there were 17 categories for apps, which are shown in Table 2.

Table 2. Dataset details.

Dataset	Service	Application
ISCX-VPN	Chat	AIM_Chat
	Email	Email
	File_Transfer	Facebook
	P2P	Gmail
	Streaming	Hangout
	VoIP	ICQ
	VPN-Chat	Netflix
	VPN-Email	SCP
	VPN-FT	Skype
	VPN-P2P	Spotify
	VPN-Streaming	Tor
	VPN-VoIP	Torrent
		Vimeo
		VoipBuster
		VPN-Ftps
		VPN-Sftp
		Youtube

3.3. Data Preprocessing

To minimize the superfluous information in the raw traffic and to enable the data to satisfy the input form of our model, we first employed the Datagram2Token tool [8] to process the raw traffic via the following procedure. We sliced the dataset's data according to the packet level and then deleted the dynamic host configuration protocol (DHCP) and address resolution protocol (ARP) packets, which did not contain relevant information about the applications that generated them. Since packet headers have robust discriminative information, such as the port number and IP address, which can cause biased inference [9], we removed the protocol ports of the TCP header, the IP header, and the Ethernet header, and we then used the bi-gram model to encode the hexadecimal sequences [8]. Finally, we randomly selected at most 5000 packets from each class. All the classes in the two tasks had a sample size of 5000, except for the application identification task, where the sample sizes of the AIM_Chat and ICQ classes were 1340 and 823, respectively. The data samples used in our model were identical to the ET-BERT [8]. The dataset was grouped into a training dataset, validation dataset, and test dataset in the proportion of 8:1:1. We preprocessed the ISCX-VPN dataset according to the characteristics of the service identification and application identification tasks. The ISCX-VPN dataset was divided into the ISCX-VPN-Service and ISCX-VPN-App datasets. Table 3 shows the statistical information of the datasets.

Table 3. The statistical information of the datasets.

Dataset	Number of Samples	Number of Labels
ISCX-VPN-Service	60,000	12
ISCX-VPN-App	77,163	17

3.4. Background on Neural Networks

3.4.1. BERT

The BERT model released by the Google AI team has made a milestone change in the field of NLP and has dramatically propelled the research on NLP forward [27]. BERT has shown remarkable performance in the Stanford Question Answering Dataset 1.1 (SQuAD) challenge [27], the top-level contest of machine reading comprehension, outperforming humans across the board in two evaluation metrics and excelling in eleven different NLP tasks. The BERT model opened a new era of NLP, and the BERT model gained popularity in many other fields as well. BERT also obtained excellent performance in the field of cyber security. The architecture of the BERT model is illustrated in Figure 2.

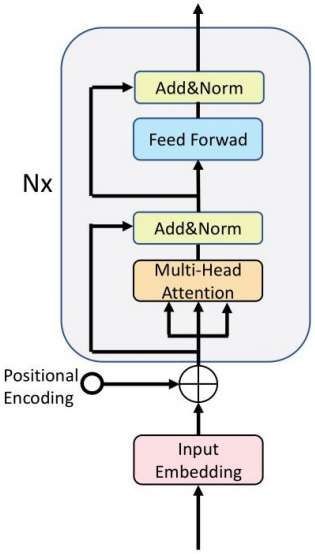

Figure 2. BERT model structure.

ET-BERT is an emerging and powerful pretrained model for encrypted traffic classification. The model learns generic packet-level representations by pretraining on a large volume of unlabeled data and then fine-tunes using a small number of task-specific labeled data. ET-BERT reached state-of-the-art performance on five encrypted traffic classification tasks. ET-BERT comprises 12 transformer blocks, each with 12 attention heads in each self-attention layer. The dimension of every input token H is set to 768, and the number of input tokens is 512 [8].

3.4.2. Convolutional Neural Network

The construction of convolutional networks is inspired by the structure of biological vision [28]; hence, convolutional neural networks (CNNs) exert superior performance in learning local features or features that are not stationary in location [1]. CNNs have excelled in various fields, including natural language processing [29] and machine vision [30]. CNNs have also shown outstanding performance in encrypted traffic in recent years; for example, using DeepPacket, the authors of [9] used CNNs to achieve a very high accuracy rate on the ISCX dataset. When convolutional networks perform image classification tasks, successive convolutional layers can extract features from individual pictures. It is observed that the features extracted in the shallow convolutional layer are relatively straightforward, such as edges and curves. The features captured in the deep convolutional layer are much more complex than those extracted in the shallow convolutional layer. The features obtained in the intermediate layers of the network are frequently complex for humans to understand. For example, in one-dimensional CNN (1D-CNN), which we applied in encrypted traffic, the feature vectors captured in the shallow layer are pointless figures that are not helpful to humans [9].

We are convinced that 1D-CNN is an optimal choice for extracting the local features of traffic. This is because 1D-CNN can capture the spatial dependencies between different bytes of packets [9], thereby offering more helpful information to the classifier and obtaining a better classification of encrypted traffic. Our model's classification performance confirms this claim.

3.4.3. Fully Connected Layer

The fully connected layer (FC) commonly serves as a classifier in deep learning models, mapping the data features learned by the model to the labeled space. The core operation of the fully connected layer is the matrix–vector product; Its formula is as follows: The symbol W refers to the weight of the fully connected layer, and symbol B refers to the bias of the fully connected layer. X and Y mean the input and output vectors, respectively.

$$Y = W \times X + B \tag{1}$$

3.5. Modules of the BFCN Model

3.5.1. Packet Encoder Layer

In this layer, we extracted the packet-level feature vectors by exploiting the attention mechanism of BERT and converted the encrypted traffic into the input vectors of the byte feature convolutional layer. We chose the ET-BERT pretrained model. It can learn prominent generic packet-level feature vectors.

At the packet encoder layer, the attention mechanism can efficiently identify the correlation between each byte in a packet; in short, attention can be interpreted as a vector parameter of significance weights [21]. In this way, the model can extract the prominent packet-level global features. The first token of each token embedding sequence is always token [CLS], and this token's relevant final hidden layer represents the packet-level global features of the input data. Then the states of the associated final hidden layers of the remaining tokens represent the features of the corresponding tokens. The states of the associated hidden layers of these tokens serve as the input to the next layer for extracting the local features.

3.5.2. Byte Feature Convolutional Layer

After processing the packet encoder layer, to be able to capture more feature information in the packet, we adopted the idea of capturing locally salient features, i.e., a one-dimensional convolution operation using multiple filters. After data preprocessing, each token comprises two adjacent bytes in the traffic. Then the final hidden layer state of each token incorporates the token information associated with it under the effect of the packet encoder layer. We took the final hidden state in the packet encoder layer except for token [CLS] as the input of this layer. After three one-dimensional convolutional filter convolution operations, we extracted a local byte-level feature vector. Finally, the global packet-level features were concatenated with the local byte-level features to acquire a feature vector containing more information and then were input into a fully connected layer for the traffic prediction task.

4. Experiment

In order to improve the performance of encrypted traffic classification, we proposed a BFCN model based on BERT and CNN. We conducted ttraffic service and application identification tasks on the ISCX-VPN dataset to evaluate our model. The experimental results are shown in this section.

4.1. Experiment Setting

This experiment was performed using Python version 3.8, with the ubuntu20.04 operating system; the processor is 14 core Intel(R) Xeon(R) Gold 6330 CPU @ 2.00 GHz, the graphics processing unit is single 3090, and the size of the graphics processing unit memory is 24 GB.

All experiments were based on PyTorch version 1.11.0 and universal encoder representations (UERs) [31]. The model's parameters were as follows: we set the maximum input length to 128 tokens, the batch size to 32, the training epochs number to 16, and the learning rate to 2×10^{-5}, and we chose AdamW [32] as the optimizer.

4.2. Evaluation Metrics

We used four evaluation metrics: accuracy (AC), precision (PR), recall (RC), and F1 score (F1). Macro-averaging [33] aims to avoid biased results due to the imbalance between multiple classes by calculating the average of each class AC, PR, RC, and F1. When evaluating the model, the category of interest was generally treated as the positive class, while other categories were treated as negative classes. *TP* is the number of positive samples correctly identified, *FP* is the number of negative samples wrongly identified, *TN* is the number of negative samples correctly identified, and *FN* is the number of positive samples wrongly identified.

$$\text{Acuuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

$$\text{Recall} = \frac{TP}{TP + TN} \tag{4}$$

$$\text{F1} - \text{score} = \frac{2 * \text{ Precision } * \text{ Recall}}{\text{Precision } + \text{ Recall}} \tag{5}$$

4.3. Effect of Sequence Length and the Number of CNN Layers

The sequence length is an essential parameter in the BERT model, which allows a sequence length of up to 512. If it is higher than the defined length, then the BERT model performs a truncation operation. If it is shorter than the defined length, the BERT model

performs a fill [PAD] token operation. If it is lower than the defined length, the BERT model performs a fill [PAD] token operation. The size of the sequence length affects the performance of the model. Too great a value of sequence length will introduce too many nonsensical [PAD] tokens in the data, and too short a sequence length will lose much information in the traffic data. Therefore, a suitable value should be chosen.

Table 4 shows the accuracy of our model at different sequence lengths. As the sequence length increased, the model's accuracy increased overall, and our model performance reached its maximum at a sequence length of 128. Therefore, we set the value of the maximum sequence length to 128.

Table 4. Effect of sequence length (accuracy).

Sequence Length	Service Identification	App Identification
8	65.83	92.83
16	98.87	99.46
32	99.02	99.02
64	98.85	99.57
128	99.12	99.65

Table 5 shows the performance of our model with various numbers of CNN layers for service identification and application identification. As the number of layers increased, the model accuracy increased, and our model reached the best performance when the number of layers was three. When the number of layers was four, the performance of the model then decreased or remained unchanged because too many layers introduce more parameters and computations, affecting the model's performance. Therefore, the number of CNN layers in our model was set to three.

Table 5. Effect of the number of CNN layers (accuracy).

Number of CNN Layers	Service Identification	App Identification
1	98.8	99.60
2	99.02	99.62
3	99.12	99.65
4	98.83	99.65

4.4. Comparison with Different Methods

4.4.1. The Benchmark Methods

In this paper, 12 of the most well-known and recognized methods in encrypted traffic classification were selected as benchmark methods. To further validate our model, we designed a BERT+BiLSTM model for comparison, extracting the time-series features using bidirectional long short-term memory (BiLSTM) on most of the input of the final hidden layer of tokens other than token [CLS]. The time series and packet features were then concatenated as the final feature representation of the traffic. The benchmark methods use data that are consistent with the data used in our methods.

1. Fingerprint construction approach: FlowPrint [6];
2. Statistical feature approaches: AppScanner [16], CUMUL [7], BIND [17], and k-fingerprinting (K-fp) [34];
3. Deep learning approaches: deep fingerprinting (DF) [18], FS-Net [19], GraphDApp [35], TSCRNN [20], and DeepPacket [9];
4. Pretraining approaches: PERT [23], ET-BERT(flow) [8], ET-BERT [8](packet), and BERT+ BiLSTM.

4.4.2. Experimental Results

Tables 6 and 7 show the experimental results of our proposed model and other models for service identification and application identification on the ISCX-VPN dataset.

Table 6. Comparison results on service identification tasks [8].

Method	Accuracy	Precision	Recall	F1-Score
AppScanner [16]	71.82	73.39	72.25	71.97
CUMUL [7]	56.10	58.83	56.76	56.68
BIND [17]	75.34	75.83	74.88	74.20
K-fp [34]	64.30	64.92	64.17	63.95
FlowPrint [6]	79.62	80.42	78.12	78.20
DF [18]	71.54	71.92	71.04	71.02
FS-Net [19]	72.05	75.02	72.38	71.31
GraphDApp [35]	59.77	60.45	62.20	60.36
TSCRNN [20]	N/A	92.70	92.60	92.60
DeepPacket [9]	93.29	93.77	93.06	93.21
PERT [23]	93.52	94.00	93.49	93.68
ET-BERT(flow) [8]	97.29	97.56	97.31	97.33
ET-BERT(packet) [8]	98.90	98.91	98.90	98.90
BERT+BiLstm	99.05	99.05	99.05	99.05
Proposed	99.12	99.13	99.11	99.11

Table 7. Comparison results on application identification tasks [8].

Method	Accuracy	Precision	Recall	F1-Score
AppScanner [16]	62.66	48.64	51.98	49.35
CUMUL [7]	53.65	41.29	45.35	42.36
BIND [17]	67.67	51.52	51.53	49.65
K-fp [34]	60.70	54.78	54.30	53.03
FlowPrint [6]	87.67	66.97	66.51	65.31
DF [18]	61.16	57.06	47.52	47.99
FS-Net [19]	66.47	48.19	48.48	47.37
GraphDApp [35]	63.28	59.00	54.72	55.58
TSCRNN [20]	N/A	N/A	N/A	N/A
DeepPacket [9]	97.58	97.85	97.45	97.65
PERT [23]	82.29	70.92	71.73	69.92
ET-BERT(flow) [8]	85.19	75.08	72.94	73.06
ET-BERT(packet) [8]	99.62	99.36	99.38	99.37
BERT+BiLstm	99.61	99.29	99.43	99.35
Proposed	99.65	99.36	99.47	99.41

In the service identification task, our method achieved the best performance in four evaluation metrics and outperformed all the baseline models. The BFCN model had the highest accuracy, precision, recall, and F1 values of 99.12%, 99.13%, 99.11%, and 99.11%, respectively. Our method improved by 5.83%, 5.36%, 6.05%, and 5.90% over the DeePacket model, using only 1D convolution in the four metrics. It improved by 0.22%, 0.22%, 0.21%, and 0.21% over the ET-BERT model on the four metrics, respectively.

In the application identification task, our method outperformed all the baseline models in terms of accuracy, recall, and F1 values, achieving the best performance. The accuracy, precision, recall, and F1 values of the BFCN model were 99.65%, 99.36%, 99.47%, and 99.41%, respectively. Our method improved by 2.07%, 1.51%, 2.02%, and 1.76% over the DeePacket model using only one-dimensional convolution in the four metrics. The improvement over the ET-BERT model was 0.03%, 0.09%, and 0.04% in accuracy, recall, and F1 values, respectively.

In the service identification task, the BERT+BiLSTM model that we designed for comparison achieved 99.05% in four evaluation metrics, which were 0.15%, 0.14%, 0.15%, and 0.15% improvement over the BERT in all four metrics. However, none of them had a better performance than the BFCN model. In terms of application recognition, compared with the ET-BERT model, there was a slight decrease in the other three metrics, except

for a slight increase in recall. The experimental comparison results show that combining packet-level global and byte-level local traffic features can better represent the traffic.

Tables 8 and 9 show the comparison between our model and the DeePacket model for each class in both tasks; DeePacket only uses a 1D convolutional network to extract local features. It is evident that our model has superior performance. Confusion matrices are commonly used to validate the classification results and provide a better view of how well the model works. The confusion matrices of the BFCN model for service identification and application identification are shown in Figures 3 and 4, respectively. The confusion matrix rows correspond to the sample's actual class, while the columns are the labels given by the model predictions.

Table 8. Performance for each type of encrypted traffic service identification.

	Proposed			**DeePacket [9]**		
Class	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Chat	97.1	98.8	97.9	84	71	77
Email	98.8	97.2	98.0	96	87	91
File_Transfer	100	99.6	99.8	98	100	99
P2P	99.8	100	99.9	100	100	100
Streaming	99.8	99.8	99.8	92	87	90
VoIP	99.4	99.8	99.6	63	88	74
VPN-Chat	99.6	99.2	99.4	98	98	98
VPN-Email	99.6	100	99.8	99	98	99
VPN-FT	99.4	96.0	97.7	99	99	99
VPN-P2P	99.8	100	99.9	100	100	100
VPN-streaming	99.8	99.8	99.8	100	100	100
VPN-VoIP	96.5	99.2	97.8	99	100	100

Table 9. Performance for each type of encrypted traffic application identification.

	Proposed			**DeePacket [9]**		
Class	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
AIM_Chat	99.2	97.8	98.5	87	76	81
Email	99.4	100	99.7	97	82	89
Facebook	99.2	99.0	99.1	96	95	96
Gmail	98.6	99.4	99.0	97	95	96
Hangout	100	99.6	99.8	96	98	97
ICQ	94.1	97.6	95.8	72	80	76
Netflix	100	100	100	100	100	100
SCP	100	99.4	99.7	97	99	98
Skype	99.4	99.8	99.6	94	99	97
Spotify	100	100	100	98	98	98
Tor	100	100	100	100	100	100
Torrent	99.2	99.8	99.5	100	100	100
Vimeo	100	100	100	99	99	99
VoipBuster	100	98.8	99.4	99	100	99
VPN-Ftps	100	100	100	100	100	100
VPN-Sftp	100	99.8	99.9	100	100	100
Youtube	100	100	100	99	99	99

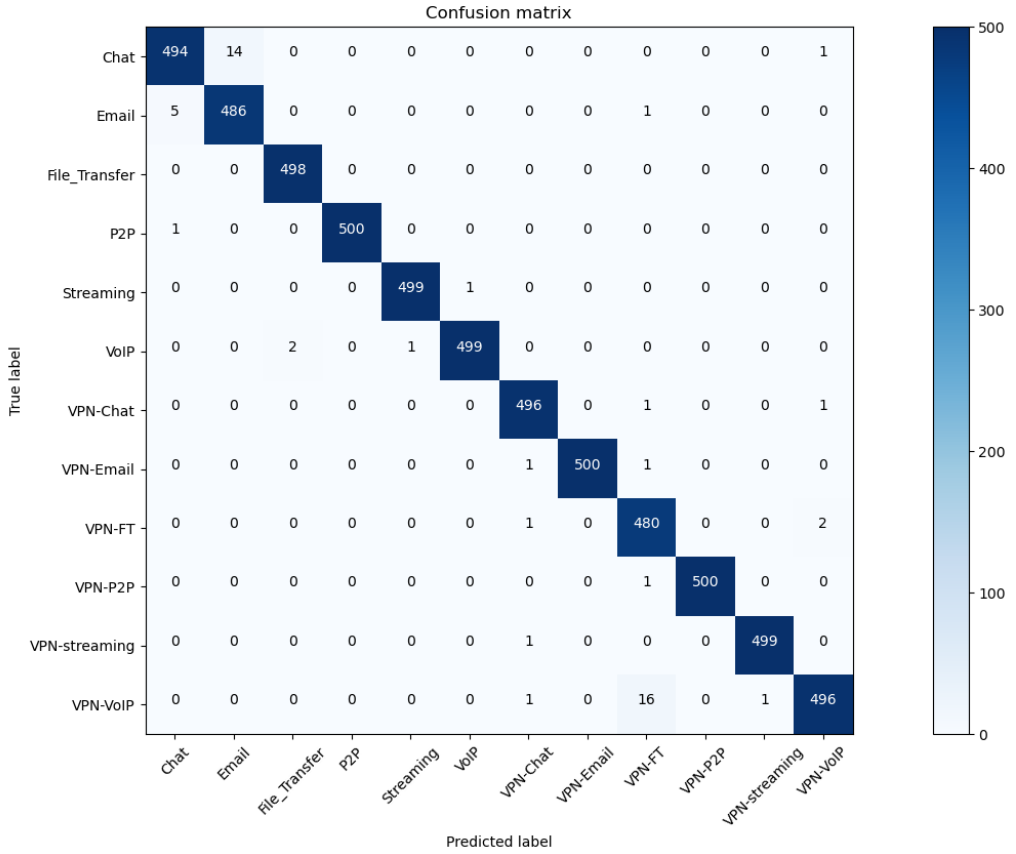

Figure 3. Confusion matrix for service identification tasks.

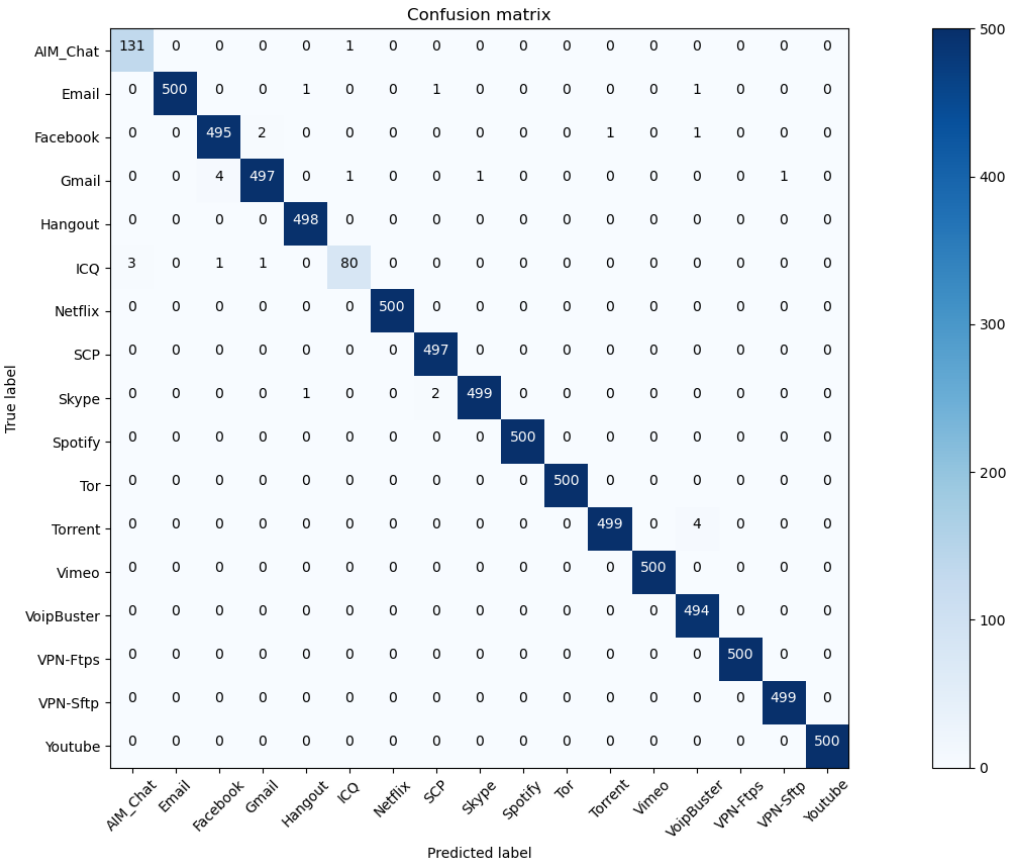

Figure 4. Confusion matrix for application identification.

After careful observation and analysis, we found that the results using only the CNN model did not result in good performance since CNN is superior at capturing local features and tends to ignore global features. In contrast, the BERT model and its variants can extract global features in the data with the advantage of a multi-headed attention mechanism; however, it ignores the salient local features of the data. Taking advantage of both can further improve the accuracy of encrypted traffic classification.

In conclusion, the experimental results demonstrate that our proposed method performs very effectively and is more competitive. It can classify encrypted traffic with high accuracy. Previous methods are affected by long sequences and a large amount of information, resulting in poor classification results. Our proposed method can effectively address this problem. In our model, the BERT-based packet encoder module extracts the packet-level global features, while the CNN-based byte feature convolution module extracts the byte-level local features, and finally, the two are fused into a final traffic feature representation for predicting the class of encrypted traffic. In future practical applications, our proposed method can significantly reduce the troubles caused by misclassification. Although our method performs very well, it also has some limitations. The primary purpose of our proposed method was to classify encrypted traffic, and it does not consider the characteristics of malicious traffic, e.g., Android software traffic. We will study more types of traffic and apply our method to this traffic model in our future research.

5. Conclusions

In this paper, we proposed a BFCN model based on BERT and CNN to improve the performance of encrypted traffic classification. The model consists of a packet encoder layer and a byte feature convolutional layer. Our approach captures packet-level global features and byte-level local features. The two feature vectors are fused as the final representation of the encrypted traffic. Finally, it is fed into the classifier to predict the corresponding class to which the encrypted traffic belongs. Experimental results show that our model achieved state-of-the-art performance for the service identification and encrypted traffic application recognition tasks on the ISCX-VPN dataset, achieving the highest accuracy rates of 99.12% and 99.65%, respectively. The improvement was 0.22% and 0.03%, compared with the baseline model. Our approach can more sufficiently exploit the features of encrypted traffic, using the packet-level global features of packets and the byte-level local features in packets. Although our approach revealed superior performance, it has also some limitations. Since we mainly considered VPN encrypted traffic, we did not consider the characteristics of other encrypted traffic and the characteristics of malicious traffic. In future research, we will apply our method to more types of traffic.

Author Contributions: Conceptualization, Z.S. and N.L.; methodology, Z.S.; software, Z.S.; validation, Z.S., Y.S. and G.T.; formal analysis, Z.S.; investigation, Y.S.; resources, Z.S.; data curation, G.T.; writing—original draft preparation, Z.S.; writing—review and editing, Z.S.; visualization, Y.S. and G.T.; supervision, N.L.; project administration, N.L.; funding acquisition, N.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded in part by the National Social Science Fund of China under Grant 20&ZD293, and as part of the Innovation Environment Construction Special 355 Project of Xinjiang Uygur Autonomous Region under Grant PT1811.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://www.unb.ca/cic/datasets/vpn.html.

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

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, Z.; Han, X.; Liu, Z.; Jiang, X.; Sun, M.; Liu, Q. ERNIE: Enhanced language representation with informative entities. *arXiv* **2019**, arXiv:1905.07129.
2. Bader, O.; Lichy, A.; Hajaj, C.; Dubin, R.; Dvir, A. MalDIST: From Encrypted Traffic Classification to Malware Traffic Detection and Classification. In Proceedings of the 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2022; IEEE: New York, NY, USA, 2022; pp. 527–533.
3. Bagui, S.; Fang, X.; Kalaimannan, E.; Bagui, S.C.; Sheehan, J. Comparison of machine-learning algorithms for classification of VPN network traffic flow using time-related features. *J. Cyber Secur. Technol.* **2017**, *1*, 108–126. [CrossRef]
4. Soleymanpour, S.; Sadr, H.; Beheshti, H. An efficient deep learning method for encrypted traffic classification on the web. In Proceedings of the 2020 6th International Conference on Web Research (ICWR), Tehran, Iran, 22–23 April 2020; IEEE: New York, NY, USA, 2020; pp. 209–216.
5. Lin, P.C.; Lin, Y.D.; Lai, Y.C.; Lee, T.H. Using string matching for deep packet inspection. *Computer* **2008**, *41*, 23–28. [CrossRef]
6. van Ede, T.; Bortolameotti, R.; Continella, A.; Ren, J.; Dubois, D.J.; Lindorfer, M.; Choffnes, D.; van Steen, M.; Peter, A. Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic. In Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 23–26 February 2020; Volume 27.
7. Panchenko, A.; Lanze, F.; Pennekamp, J.; Engel, T.; Zinnen, A.; Henze, M.; Wehrle, K. Website Fingerprinting at Internet Scale. In Proceedings of the NDSS, San Diego, CA, USA, 21–24 February 2016.
8. Lin, X.; Xiong, G.; Gou, G.; Li, Z.; Shi, J.; Yu, J. ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification. In Proceedings of the ACM Web Conference 2022, Lyon, France, 25–29 April 2022; pp. 633–642.
9. Lotfollahi, M.; Jafari Siavoshani, M.; Shirali Hossein Zade, R.; Saberian, M. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Comput.* **2020**, *24*, 1999–2012. [CrossRef]
10. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
11. Wang, W.; Zhu, M.; Wang, J.; Zeng, X.; Yang, Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, China, 22–24 July 2017; IEEE: New York, NY, USA, 2017; pp. 43–48.
12. Bhatia, M.; Sharma, V.; Singh, P.; Masud, M. Multi-level P2P traffic classification using heuristic and statistical-based techniques: A hybrid approach. *Symmetry* **2020**, *12*, 2117. [CrossRef]
13. Dainotti, A.; Pescape, A.; Claffy, K.C. Issues and future directions in traffic classification. *IEEE Netw.* **2012**, *26*, 35–40. [CrossRef]
14. Qi, Y.; Xu, L.; Yang, B.; Xue, Y.; Li, J. Packet classification algorithms: From theory to practice. In Proceedings of the IEEE INFOCOM 2009, Rio de Janeiro, Brazil, 19–25 April 2009; IEEE: New York, NY, USA, 2009; pp. 648–656.
15. Madhukar, A.; Williamson, C. A longitudinal study of P2P traffic classification. In Proceedings of the 14th IEEE international symposium on modeling, analysis, and simulation, Monterey, CA, USA, 11–14 September 2006; IEEE: New York, NY, USA, 2006; pp. 179–188.
16. Taylor, V.F.; Spolaor, R.; Conti, M.; Martinovic, I. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 63–78. [CrossRef]
17. Al-Naami, K.; Chandra, S.; Mustafa, A.; Khan, L.; Lin, Z.; Hamlen, K.; Thuraisingham, B. Adaptive encrypted traffic fingerprinting with bi-directional dependence. In Proceedings of the 32nd Annual Conference on Computer Security Applications, Los Angeles, CA, USA, 5–9 December 2016; pp. 177–188.
18. Sirinam, P.; Imani, M.; Juarez, M.; Wright, M. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1928–1943.
19. Liu, C.; He, L.; Xiong, G.; Cao, Z.; Li, Z. Fs-net: A flow sequence network for encrypted traffic classification. In Proceedings of the IEEE INFOCOM 2019-IEEE Conference On Computer Communications, Paris, France, 29 April–2 May 2019; IEEE: New York, NY, USA, 2019; pp. 1171–1179.
20. Lin, K.; Xu, X.; Gao, H. TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of IIoT. *Comput. Netw.* **2021**, *190*, 107974. [CrossRef]
21. Chen, X.; Cong, P.; Lv, S. A Long-Text Classification Method of Chinese News Based on BERT and CNN. *IEEE Access* **2022**, *10*, 34046–34057. [CrossRef]
22. Sengupta, S.; Ganguly, N.; De, P.; Chakraborty, S. Exploiting diversity in android tls implementations for mobile app traffic classification. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 1657–1668.
23. He, H.Y.; Yang, Z.G.; Chen, X.N. PERT: Payload encoding representation from transformer for encrypted traffic classification. In Proceedings of the 2020 ITU Kaleidoscope: Industry-Driven Digital Transformation (ITU K), Ha Noi, Vietnam, 7–11 December 2020; IEEE: New York, NY, USA, 2020; pp. 1–8.
24. Hu, X.; Gu, C.; Chen, Y.; Wei, F. CBD: A deep-learning-based scheme for encrypted traffic classification with a general pre-training method. *Sensors* **2021**, *21*, 8231. [CrossRef] [PubMed]
25. Jia, K. Sentiment classification of microblog: A framework based on BERT and CNN with attention mechanism. *Comput. Electr. Eng.* **2022**, *101*, 108032. [CrossRef]

26. Draper-Gil, G.; Lashkari, A.H.; Mamun, M.S.I.; Ghorbani, A.A. Characterization of encrypted and vpn traffic using time-related. In Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), Rome, Italy, 19–21 February 2016; pp. 407–414.
27. Rogers, A.; Kovaleva, O.; Rumshisky, A. A primer in bertology: What we know about how bert works. *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 842–866. [CrossRef]
28. Hubel, D.H.; Wiesel, T.N. Receptive fields and functional architecture of monkey striate cortex. *J. Physiol.* **1968**, *195*, 215–243. [CrossRef] [PubMed]
29. Dos Santos, C.; Gatti, M. Deep convolutional neural networks for sentiment analysis of short texts. In Proceedings of the COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, Dublin, Ireland, 23–29 August 2014; pp. 69–78.
30. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
31. Zhao, Z.; Chen, H.; Zhang, J.; Zhao, X.; Liu, T.; Lu, W.; Chen, X.; Deng, H.; Ju, Q.; Du, X. UER: An Open-Source Toolkit for Pre-training Models. In Proceedings of the EMNLP-IJCNLP 2019, Hong Kong, China, 3–7 November 2019; p. 241.
32. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
33. Liu, C.; Wang, W.; Wang, M.; Lv, F.; Konan, M. An efficient instance selection algorithm to reconstruct training set for support vector machine. *Knowl.-Based Syst.* **2017**, *116*, 58–73. [CrossRef]
34. Hayes, J.; Danezis, G. k-fingerprinting: A robust scalable website fingerprinting technique. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; pp. 1187–1203.
35. Shen, M.; Zhang, J.; Zhu, L.; Xu, K.; Du, X. Accurate decentralized application identification via encrypted traffic analysis using graph neural networks. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 2367–2380. [CrossRef]