# CUFT: Cuflow-Based Approach with Multi-headed Attention Mechanism for Encrypted Traffic Classification

Xin Zong, Min Luo(✉), Cong Peng, and Debiao He

Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China
{zongxin,mluo,cpeng,hedebiao}@whu.edu.cn

**Abstract.** Encrypted traffic classification is essential to network optimization, quality of service improvement, and network security maintenance. However, this task has become increasingly challenging with the emergence of new applications and protocols. Existing deep learning-based methods either directly use the raw packet payloads for classification without considering the contextual relationships between packets, or use flow's statistical features ignoring the inline relationships between packet payloads. Consequently, these approaches result in less accurate or less generalizable classification, which cannot be adapted to different classification scenarios. To address them, we propose a method called CUFT, which involves separating the irrelevant packets from the original traffic and extracting the continuous unidirectional flows (cuflows) for classification. Subsequently, we utilize the long-distance capturing capability of the multi-head attention mechanism to obtain the contextual relationships among the relevant packets in the cuflow and the inline relationships among the packet payloads. To verify the effectiveness and generality of CUFT across various classification tasks, we conducted experiments on three complex classification tasks using three publicly available datasets, ISCX-Tor, ISCX-VPN and USTC-TFC. Our results show that our proposed method achieves exceptional accuracy rates of 99.67%, 98.33% and 99.18% on these three complex classification tasks respectively.

**Keywords:** Encrypted traffic classification · Cuflow · Multi-head attention

## 1 Introduction

Network traffic classification is a method to classify traffic into appropriate categories according to different needs. Traffic classification with high recognition accuracy and fine granularity can provide reliable and effective technical support for network managers and optimize the network configuration to provide


H. Duan et al. (Eds.): SecureComm 2023, LNICST 567, pp. 23–40, 2025.
https://doi.org/10.1007/978-3-031-64948-6_2

better service quality based on user behaviour [1]. People are expecting more and more privacy and anonymity online as a result of data leaks, network intrusions, and ransomware. Encrypted traffic technologies and anonymization tools are widely used to achieve different levels of privacy protection. On the other hand, some cybercriminals are able to encrypt the transmission content and hide personal information such as the source and destination of the communication using technologies like Tor and VPN. As a result, it is quite challenging to track their information in the network [2].

Encrypted traffic classification research has evolved to address the above challenges over time. Conventional port-based and Deep Packet Inspection (DPI) techniques are no longer applicable for contemporary traffic classification owing to the prevalence of port and protocol obfuscation mechanisms. Machine learning has achieved remarkable results in many fields, inspiring research scholars in traffic analysis to analyze encrypted traffic from a statistical perspective. Researchers construct manual feature sets based on expert knowledge and analytical judgments of different types of traffic and then typically employ classical machine learning algorithms for classification. The design of feature sets has a significant impact on the outcomes of classification tasks. In practical scenarios, researchers who curate datasets based on their expertise may occasionally inadvertently omit crucial features, consequently influencing the outcomes of classification experiments [3]. The high reliance on manually extracted features limits their generality and does not cope well with different types of encrypted traffic.

Deep learning has a higher learning ability to learn more complex and variable patterns. Therefore, the applied research of deep learning has received more and more attention. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are often used alone or in combination for traffic classification with good results [4–7]. However, most of the above methods use each packet's part of payload as an independent sample input to the neural network, only considering the inline relationships for a single packet's payload. Based on the temporal correlation observed in network traffic, it is common to find interrelated patterns among multiple consecutive packets generated within a brief time interval. However, certain methods [8–10] for feature extraction utilize consecutive packets as a flow sequence without considering their inherent correlations. i.e., some packets are used to establish connections, while some are used to transmit encrypted data. These packets are not directly related, and putting them together as input can affect the classification effect. Besides, they are based on the statistical characteristics of the flow, while the inline relationships between payloads may be ignored.

To address the above issues, we propose a cuflow-based network traffic classification scheme called CUFT (Continuous Unidirectional Flow based Transformer), which separates unrelated packets and fully explores the contextual relationships between related packets, as well as the inline relationships between payloads. We sort packets by time, group packets with time intervals less than a certain threshold into the same burst to separate unrelated packets, and extract

packets in each burst that are time-adjacent and have exactly the same quintet as a continuous unidirectional flow. We use the represented cuflow as the input and use the long-range feature capture capability of the multi-headed attention mechanism proposed from Transformer [11] to find the positional relationship between payloads and the contextual relationship between packets and packets (see Methods for details). The main contributions of our work are threefold:

(1) We propose an encrypted traffic classification method called CUFT, which can fully explore not only the inline relationships between payloads but also the relationships between different packets' contexts. It automatically extracts features without the need for manual feature engineering.
(2) In the data processing stage, we propose the extraction and representation methods of cuflow to improve the accuracy of model recognition. Segmenting the original traffic minimizes the impact of extraneous packets on classification in contrast to working with flows. Moreover, restructuring pertinent packets, as opposed to treating them individually, enables the resultant features to encompass a richer set of discriminative information, consequently enhancing classification performance.
(3) To assess the efficacy and applicability of our proposed method, we devised three intricate classification tasks employing publicly accessible datasets: ISCX-Tor (16-class), ISCX-VPN (10-class), and USTC-TFC (20-class). We conducted comparative analyses against a range of machine learning and deep learning approaches. The experimental outcomes demonstrate that CUFT consistently outperforms these methods across all three classification scenarios, manifesting a substantial performance improvement in contrast to both traditional machine learning and deep learning methods.

## 2 Related Work

In recent years, traffic analysis has primarily revolved around distinct objectives, including Quality of Service (QoS) provisioning, malware detection, and user behavior analysis. To extract the characteristics of the traffic and construct classifiers to classify the traffic, three main types of methods are commonly used: traditional methods, machine learning methods and deep learning methods.

### 2.1 Traditional Methods

Traditional methods are rule-based classification methods, mainly divided into two types: port-based and deep packet inspection (DPI). The port-based classification method extracts the packet's port and compares it with the port number assigned by IANA. However, the proliferation of port obfuscation, network address translation (NAT) and random port assignment has dramatically reduced this classification method's effectiveness. DPI is a method that classifies data messages by probing their contents in the application stream. This approach remains valid for traffic after port obfuscation. Still, it is computationally expensive and can be easily circumvented by encryption or encapsulation.

### 2.2 Machine Learning Methods

Machine learning algorithms typically use statistical features as input and are trained to distinguish different traffic. They have a biased assumption that different types of encrypted traffic contain some statistical features unique to that class [12]. Naami et al. [13] extracted temporal statistical features of upstream and downstream packets from the header and introduced an adaptive model to train a support vector machine (SVM); Anderson et al. [14] proposed to combine 22 standard datasets [15] and an augmented dataset containing TLS Handshake Metadata and incorporated as a traffic feature set to distinguish 18 malicious traffic. The above methods only extract features and don't select the extracted features. Shen et al. [16] performed feature selection on the extracted features and then fused them using kernel functions to distinguish different dapps. These approaches heavily rely on features that are designed by humans. Designing universally applicable statistical features capable of addressing diverse classification tasks within the context of progressively intricate traffic types poses a significant challenge.

### 2.3 Deep Learning Methods

Deep learning algorithm has better robustness and feature extraction capability and has been applied to network traffic analysis in complex environments [17]. FlowPic [9] plotted a two-dimensional histogram using packet arrival time and packet size as input to the model. Deeppacket [12] intercepted the first 1500 bytes of each packet, the insufficient ones were filled with zeros and normalized as the input of the model, and then two types of encrypted traffic were classified using two neural networks, stacked autoencoder and CNN. TSCRNN [18] used 15 packets, each intercepting 1500 bytes as input, then used CNN to extract spatial features and introduced LSTM to learn temporal features to classify Tor-encrypted traffic. Cheng et al. [8] proposed an online encrypted classification method, which used the statistical features of three consecutive data packages as model inputs and then extracted features using a multi-headed attention mechanism to perform classification on the three datasets. Lin et al. [19] proposed a novel traffic representation model that can be trained for different classification tasks by fine-tuning on a small amount of task-specific labeled data.

The aforementioned deep learning methods are generally divided into two categories, one based on raw packets' payloads and the other on flow-based statistical features. If employing packets' payloads for classification purposes, the interconnections among packets might be disregarded. Meanwhile, when utilizing the statistical attributes of flows, unrelated packets could potentially disrupt the outcomes, leading to the neglect of inherent sequential associations within packets' payloads. Therefore, we use the cuflows' payloads for classification to reduce the interference of irrelevant packets on the results and also make the model focus on the contextual relationships between relevant packets and the inline relationships between packets' payloads.

# 3 Methodology

In this work, we propose a traffic classification scheme called CUFT, consisting of two phases: data pre-processing and classification. Figure 1 shows the framework of CUFT. In the data pre-processing stage, the raw traffic data is firstly partitioned into multiple bursts. Then the cuflows are extracted from each burst. Finally, the extracted cuflows are represented and embedded to satisfy the neural network's input. In the classification phase, the neural network will mine the multidimensional dependencies inherent in encrypted traffic, capturing the contextual associations between payloads and the sequential relationships between locations to distinguish traffic types.

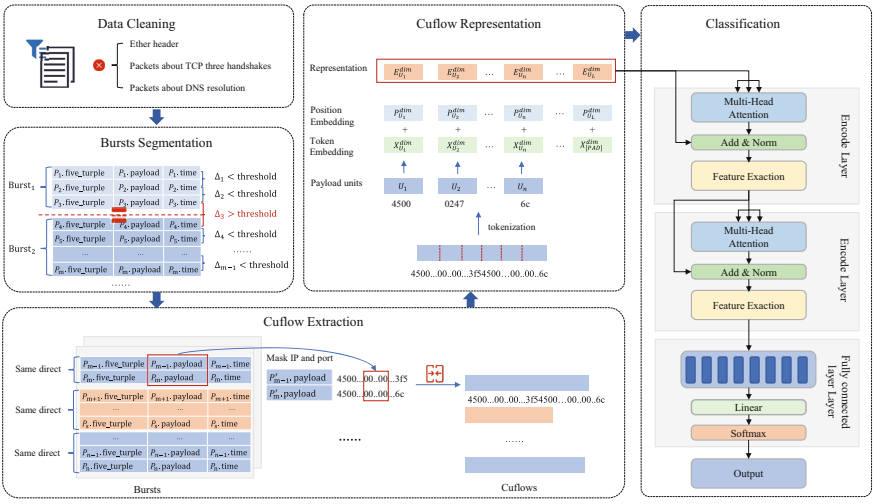


**Fig. 1.** The Framework of CUFT.

## 3.1 Data Pre-processing

Unlike natural language and images, encrypted traffic is typically encrypted using various encryption methods, resulting in packets that do not explicitly enclose semantic units and are unintelligible without the correct decryption key. The traffic needs to be transformed into distinguishable expressions to distinguish huge amounts of raw data. To this end, we propose four steps for the data pre-processing phase: (1) Data Cleaning is processing datasets to remove classification-irrelevant data to reduce interference with classification. (2) Burst Segmentation is used to separate unrelated packets from the original data traffic so that it can be transformed into multiple manageable parts for further processing. (3) Cuflow Extraction is the extraction of consecutive and identically

oriented packets from bursts in chronological order. (4) Cuflow Representation converts the cuflows' raw traffic data into a sequence of embedded tokens to satisfy the input format of the neural network model.

**Data Cleaning.** In this work, we use the public traffic datasets ISCX-Tor [20], ISCX-VPN [21] and USTC-TFC [21], which are captured in a real-world simulation. As a result, these datasets will contain information that is irrelevant to encrypted traffic, which increases model training time, interferes with encrypted traffic classification, and impacts predicted results, demanding data cleaning. The source and destination MAC addresses in the Ethernet header are only related to the transmitting and receiving devices, which do not help classify the encrypted traffic and even interfere with the prediction. Packets generated during the three TCP handshakes, or packets that establish or cancel connections, should also be removed. This is because, at this stage, packets do not include payloads or convey relevant information regarding the specific application or service type, making them irrelevant to the classification of encrypted traffic.

**Burst Segmentation.** A burst of encrypted traffic is defined as a collection of network packets captured within a temporal threshold. That is, given a period, if the interval between two time-adjacent packets is less than a certain threshold, they are considered to belong to the same burst; otherwise, they belong to different bursts. As the number of personalized services on the web increases, the Document Object Model (DOM) trees become more complex and varied. During the browser rendering prtocess, the network traffic is divided into distinct fragments that are semantically meaningful according to the type of service (text, images, videos, etc.). Burst almost contains the complete information content of this semantic fragment; thus, we divide the original traffic into multiple semantic-aware bursts to separate unrelated packets.

When considering bursts, our focus centers on the source, destination, and arrival time of messages. The definition of a burst is as follows:

$$burst = \{packet_i, i \in N^+\} \tag{1}$$

where *packet* is composed of six tuples $\{time, sip, dip, sp, dp, protocal\}$, representing the packet's timestamp, source ip address, source port, destination ip address, destination port, and application layer protocol. $\forall i, i \in N^+, packet_{i+1}.time - packet_i.time < threshold.$

**Cuflow Extraction.** Each burst comprises a bidirectional flow from the client and the server. We define a continuous unidirectional flow (cuflow) as a sequence of time-adjacent packets that share the same five-tuple properties. These cuflows can be further classified into requests sent by the client to the server or responses from the server to the client. We define the direction of packets transmitted from a client to a server as outbound and the direction of packets sent from a server to a client as inbound. As shown in Fig. 2, each cuflow contains at least one

packet with the format of $\{P_1, P_2, ..., P_m\}$, where $m$ is the maximum number of consecutive packets in the same direction. After extracting the cuflows, the packets' payloads in the cuflow need to be merged. It is worth noting that the IP address and port of the packet are highly identifiable and will affect the classifier's judgment when classifying, making a high result. Therefore, we have to mask the IP address and port of the packets before merging the packets in the cuflow.

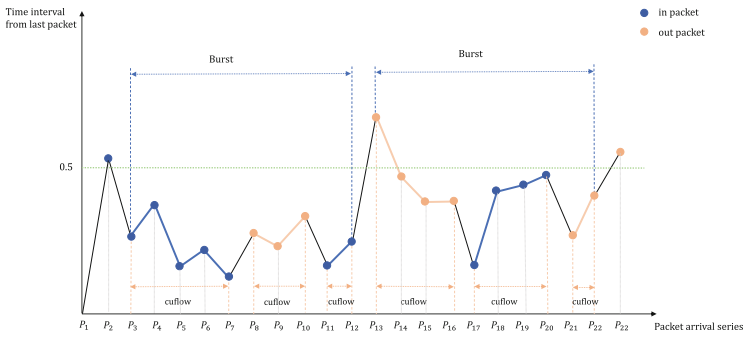


**Fig. 2.** Visualization of cuflows.

**Cuflow Representation.** The raw data of a cuflow is composed of a binary sequence. A common approach is to convert it into bytes, with a range of values from 0 to 255, which is advantageous as it is suitable to generate grayscale images and then can be used as inputs to convolutional neural networks. However, due to the limited range of intercepted traffic information, such images cannot adequately represent the overall traffic characteristics. Tokenization can be used to separate the data into distinct payload units. If we use 1 byte as the unit for slicing raw traffic, the max dictionary size is 256, which is too small to represent the characteristic information of raw traffic well. So we use double bytes as the payload unit of raw traffic, so that the max size of the traffic dictionary can be expanded to 65536. We also add $<UNK>$ for low-frequency units or units not in the dictionary and $<PAD>$ for complementary characters. We transform the cuflow into a token sequence based on the constructed traffic dictionary.

After token serialization, we use payload unit token embedding and position embedding to convert a unit vector with a fixed size. Under the random initialization function $\Phi : N \rightarrow \mathbb{R}^{dim}$, each payload unit it contains takes a random value in the normal distribution N(0,1) and the dimension of the embedding is 512. That is, $cuflow = \{U_1, U_2, ..., U_n, n \in N^+\}$, where $U_i$ is $i$th playload unit in cuflow, the vector after unit token embedding $X_c = [\Phi(U_1), \Phi(U_2), ..., \Phi(U_n)]^T = [X_{U_1}^{dim}, X_{U_2}^{dim}, ..., X_{U_n}^{dim}]^T \in \mathbb{R}^{n \times dim}$, dim is set to 512.

The raw data results from layers of encapsulation and thus contains a number of fixed fields implicitly included in each payload unit that is often closely related

to their context. To enable the model to focus on the sequential temporal relationship between each payload unit, position embedding is used to inject payload units order information to enhance the model input. The vector after position embedding $P_c = [f(U_1), f(U_2), ..., f(U_n)]^T = [P_{U_1}^{dim}, P_{U_2}^{dim}, ..., P_{U_n}^{dim}]^T \in \mathbb{R}^{n \times dim}$, where

$$f(U_t)^{(i)} = \begin{cases} sin(\frac{1}{10000^{2k/d}}, i), & i = 2k \\ sin(\frac{1}{10000^{2k/d}}, i), & i = 2k+1 \end{cases} \tag{2}$$

$E_{U_t} = \Phi(U_t) + f(U_t)$ is $i$th payload unit's final embedded vector, and the final representation of cuflow is $E_c = [E_{U_1}^{dim}, E_{U_2}^{dim}, ..., E_{U_n}^{dim}]^T \in \mathbb{R}^{n \times dim}$.

### 3.2 Classification

The nature of encrypted communication differs depending on the type of transmission element (e.g., video, image, text) and the transmission order. Network packets are transmitted using TCP segmentation and IP fragmentation techniques to successfully lower the loss rate of Internet packets, and increase the speed of data transmission by more effectively utilizing network resources. Due to the maximum segment size (MSS) and maximum transmission unit (MTU), the transmission content is divided into numerous packets for transmission. A cuflow extracted from network traffic contains multiple packets, which are usually sequential and make up a relatively complete transmission. In order to distinguish different transmission contents and to mine the contextual relationships between packets and the internal dependencies between message payloads, a spatiotemporal representation needs to be extracted from the cuflow. Inspired by the network structure of Transformer [11], a sequence learning model based on attention mechanism, we use a multi-headed self-attentive network to mine the multidimensional contextual relationships between packets within a cuflow and the internal dependencies between payloads to dynamically generate attention weights at different locations in the input cuflow. We use a streamlined structure consisting of encode layers and fully connected layers.

**Encode Layer.** The encoding layer contains two key sub-layers: the self-attention layer and the feature extraction layer. The self-attention mechanism directly calculates the attention weights for each location of the cuflow, and then calculates the implicit vector representation of the whole cuflow in the form of the sum of the weights. The parallel computation of shared weights by multi-head attention greatly reduces the number of parameters, shortens the running time, and improves efficiency. In order to extract deeper features, two linear transformations are performed in the feature extraction layer. Take a cuflow's embedding vector $E_c = [E_{U_1}, E_{U_2}, ..., E_{U_n}]^T \in \mathbb{R}^{n \times dim}$ as an example, there are several steps of the encode layer as follows:

*Linear Transformation:* The self-attention layer employs $h$ attention heads. As for attention head $i$, using linear transformation to obtain Query Matrix, Key Matrix and Value Matrix:

$$Q_i = E_c W_i^Q = [q_1, q_2, ..., q_n]^T \in \mathbb{R}^{n \times d_k} \tag{3}$$

$$K_i = E_c W_i^K = [k_1, k_2, ..., k_n]^T \in \mathbb{R}^{n \times d_k} \tag{4}$$

$$V_i = E_c W_i^V = [v_1, v_2, ..., v_n]^T \in \mathbb{R}^{n \times d_k} \tag{5}$$

where $q_i, k_i, v_i$ represent the query vector, key vector and value vector corresponding to $E_{U_i}$, and $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{dim \times d_k}$ are trainable parameter matrix. The similarity matrix between payload units can be obtained by inner product:

$$S_i = Q_i K_i^T = [q_1^T K_i^T, q_2^T K_i^T, ..., q_n^T K_i^T] = [s_1, s_2, ..., s_n] \tag{6}$$

where $s_j$ is similarity vector between the $j$th payload unit and others.

*New Representation Acquisition:* The normalized weight matrix is obtained after using the activation function Softmax to make the sum 1. And then multiplying it with the payload unit vector matrix to obtain $i$th attention head's output: $head_i = Softmax(\frac{S_i}{\sqrt{d_k}})V_i$. The output of multiple attention heads was combined and linearly transformed to create the new representation of a cuflow:

$$Z_c = Concat(head_1, head_2, ..., head_h)W^O \in \mathbb{R}^{n \times dim} \tag{7}$$

where $W^O \in \mathbb{R}^{h \times d_k \times dim}$ is parameter matrix for linear transformation to convert the dimension to cuflow's original embedding dimension.

*Feature Extraction:* The new representation obtained by the multi-headed attention mechanism globally aggregates the information of the cuflow. The feature extraction layer is to perform nonlinear transformations of the features at each time sequence to improve the network representation. When feature extraction is performed, the first-dimensional expansion operation combines the features with various features to fully explore the connection between payload units and improve the model discrimination. Then a dimensional reduction operation is performed to remove the combined features with low differentiation to obtain the final retained features. Finally, these features will be flattened as the encode layers' output, which will be input to the next layer.

**Fully Connected Layer.** The fully connected layer converts the output of the features from the encoding layer into probability values for the labels. Normalize the output of the encode layer:

$$E_O = [E_1, E_2, ..., E_n] \in \mathbb{R}^{n \times dim} \tag{8}$$

And then using the Softmax function:

$$softmax(E_O) = \frac{exp(E_i)}{\sum_{j=1}^{n} exp(E_j)} (i = 1, \cdots, n) \tag{9}$$

The crossover loss function is used as the loss function. For $N$ cuflows of $M$ classes, the loss function is calculated as follows.

$$Loss = \frac{1}{N}\sum_{i=1}^{N} Loss_i = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{M} y_{ic}\log(P_{ic}) \tag{10}$$

where $y_{ic} \in \{0, 1\}$ is the label value of the cuflow $i$ corresponding to the prediction category $c$. $P_{ic} \in (0, 1)$ is the probability that the predicted sample belongs to category $c$.

## 4 Experiment

### 4.1 Setup

**Experimental Environment.** The experimental hardware is composed of an Intel(R) Core(TM) i5-10500 CPU @ 3.10 GHz, 16G of RAM and an NVIDIA GeForce RTX 3060 GPU gas pedal. The operating system used is Ubuntu 18.04LTS. The framework runs on Python 3.8 and the deep learning platform is Pytorch 1.12.0+cu113.

**Table 1.** Description of three different classification tasks.

| Experiment | Dataset | Label | Train | Test | Label | Train | Test |
|---|---|---|---|---|---|---|---|
| A | ISCX-Tor | Audio | 410 | 111 | Tor:Audio | 337 | 84 |
| | | Browsing | 972 | 240 | Tor:Browsing | 697 | 168 |
| | | Chat | 1069 | 290 | Tor:Chat | 1025 | 277 |
| | | FT | 329 | 80 | Tor:FT | 349 | 79 |
| | | Email | 327 | 77 | Tor:Email | 464 | 125 |
| | | P2P | 497 | 116 | Tor:P2P | 456 | 124 |
| | | Video | 702 | 187 | Tor:Video | 354 | 83 |
| | | VoIP | 838 | 187 | Tor:VoIP | 775 | 171 |
| B | ISCX-VPN | Chat | 2327 | 591 | VPN:Chat | 1533 | 399 |
| | | Email | 761 | 197 | VPN:Email | 804 | 226 |
| | | FT | 3544 | 896 | VPN:FT | 298 | 81 |
| | | Streaming | 3262 | 770 | VPN:Streaming | 411 | 101 |
| | | VoIP | 3296 | 791 | VPN:VoIP | 795 | 205 |
| C | USTC-TFC | BitTrt | 545 | 138 | Cridex | 516 | 132 |
| | | Facetime | 581 | 156 | Geodo | 1741 | 444 |
| | | FTP | 665 | 153 | Htbot | 1563 | 456 |
| | | Gmail | 1012 | 288 | Miuref | 591 | 159 |
| | | MySQL | 534 | 120 | Neris | 441 | 93 |
| | | Outlook | 526 | 132 | Nsis | 1170 | 393 |
| | | Skype | 929 | 201 | Shifu | 341 | 75 |
| | | SMB | 877 | 201 | Tinba | 97 | 6 |
| | | Weibo | 2159 | 558 | Virut | 760 | 81 |
| | | Wow | 702 | 138 | Zeus | 306 | 90 |

**Dataset and Classification Tasks.** To verify our method's good generality and effectiveness for different types of traffic in different scenarios, we set three types of traffic classification tasks on three datasets. We first divide each dataset into a training set and a test set in the ratio of 8:2, where the training set is used to train the model and the test set is used to evaluate the performance of the trained model. When training the model, we divide the dataset into two parts according to 8:2 for the training dataset and the optimization parameters respectively. The labels set for each dataset and the number of samples in the training and test sets are detailed in Table 1, where a cuflow is considered as a sample.

ISCX-Tor [20] is a labeled dataset of Tor traffic published by UNB (University of New Brunswick). This dataset contains two sets of data, one is regular encrypted traffic and the other is encrypted traffic encrypted by Tor. Tor-encrypted traffic hides key information such as network structure and addresses of both sides of the communication, increasing the classification challenge. For this dataset, we set a 16-class application classification task. This task aims to classify the type of upper-layer services corresponding to the mixed encrypted traffic.

ISCX-VPN [21] is also a labeled dataset containing regular encrypted traffic and VPN encrypted traffic published by UNB. VPN is a private overlay network between distributed sites by tunneling traffic over a public communication network (tunneling). To classify the service types to which the mixed encrypted traffic belongs, we set a 10-class service type classification task.

USTC-TFC [21] has two components, one is normal benign encrypted traffic and the other is malicious traffic. We mix the two types of encrypted traffic data together and set a 20-class application classification task, aiming to distinguish the applications they belong to.

**Evaluation Metrics.** All three of our classification tasks are multiclassification tasks that macro average [22] is used to avoid bias in results due to imbalance between data from multiple categories. We use four recognized metrics: accuracy, precision, recall-score and F1-score, when performing a multi-classification task, one of the classes can be considered as binary classification with the other classes to obtain the precision and recall of each class:

$$accuracy = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \tag{11}$$

$$macro - precision = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FP_i} \tag{12}$$

$$macro - recall = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FN_i} \tag{13}$$

$$macro - F1 = \frac{1}{C}\sum_{i=1}^{C}\frac{2TP_i}{2TP_i + FP_i + FN_i} \tag{14}$$

where $C$ is the number of categories in the classification task, TP is true positive, FP is false positive, TN is true negative, and FN is false negative.

### 4.2 Hyperparameters Comparision

We have four main parameters: the threshold value of burst segmentation $T$, the maximum number of payload units for cuflows $P$, the number of heads $M$ and the number of encode layers $N$. In our experiments, we count the time intervals of adjacent packets in the three datasets as shown in Fig. 3. Since both ISCX-Tor and ISCX-VPN contain regular encrypted traffic, the datasets are divided into five categories. The horizontal coordinates are the values of the intervals in which the adjacent packets are located (it is worth noting that the last interval refers to the time interval over 4.5 s), while the vertical coordinate is the frequency value. It is found 90% of the packets will be sent within 0.5 s of the last packet sent or received. In order to reduce the parameters and computational effort of the model and the limitations of our computer hardware, we need to set the maximum of payload units for cuflows. We count the number of packets in different cuflows in different datasets and plot the cumulative frequency to observe the percentage of cuflows with the number of packets less than a certain value, as shown in Fig. 4, where the horizontal coordinate indicates the number of packets in the cuflows and the vertical coordinate indicates the cumulative frequency. It can be seen that more than 80% of cuflows on all data sets contain less than 4 packets. Since the packet size is affected by the maximum transmission unit, and each packet size usually does not exceed 1500 bytes, more than 80% of the cuflows have a payload size less than 6000 bytes. The payload unit we set is composed of double bytes, in other words, the number of payload units in more than 80% of cuflows in all data sets is no more than 3000. According to the experimental results, We use $T = 0.5$, $P = 3000$, $M = 2$, and $N = 2$ as default hyperparameters.

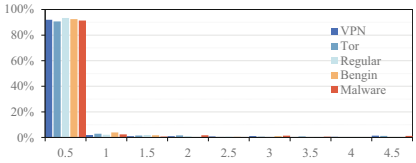


**Fig. 3.** The percentage of the number of packets contained in the time interval between different adjacent packets on different datasets.

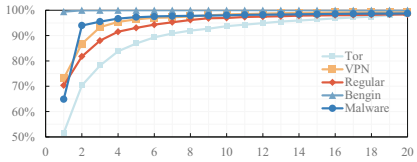


**Fig. 4.** Cumulative frequency values of the number of packets contained in cuflow for different data sets.

For each experiment, we change only one parameter and keep the default values for the other parameters. And the results of our experiments were obtained as the average of three tasks. According to Fig. 5(a), it can be seen that the performance of the model is optimal when $T = 0.5$. This is because if $T$ is too small, the number of packets in each cuflow is small, so the contextual relationship between packets cannot be extracted well. If $T$ is too large, the cuflow will contain irrelevant packets, interfering with the classification results. In Fig. 5(b), we find that the accuracy reaches 98.26% when $P = 1500$, and as $P$ increases, the accuracy increases but not more than 1%, which means 1500 payload units is enough to represent cuflow. In Fig. 5(c), we find that the performance of the model is optimal when the number of multi-heads is 4. When increasing the number of multi-heads, the performance of the model decreases. According to Fig. 5(d), when $N = 2$, the accuracy of our model reaches 98.33%. After increasing the value of $N$, the accuracy rate will no longer improve significantly. This indicates that the model with two encode layers is able to extract the inline and contextual relationships of packets well. The hyperparameters' description and the corresponding values' selection are detailed in Table 2.

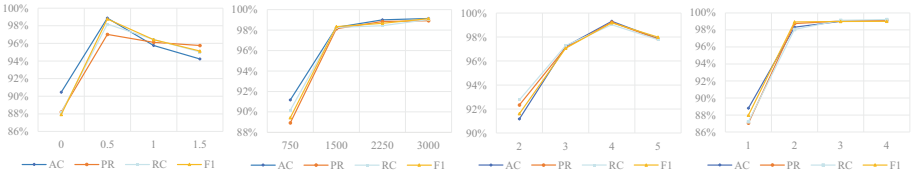


**Fig. 5.** Comparison of the effect of different hyperparameters on the results. (a) The threshold value; (b) Number of payload units; (c) Number of headers; (d) Number of Encode layer.

**Table 2.** The selected parameters of CUFT.

| Description | Parameter | Value |
|---|---|---|
| Threshold value of burst segmentation | T | 0.5 |
| Maximum number of payload units for cuflows | P | 1500 |
| Number of heads | M | 4 |
| Number of encode layers | N | 2 |
| The number of cuflow selected for one training | batch_size | 32 |
| Dimension of cuflow embedding | dim | 512 |
| Dimension of feature extraction layer | dim_e | 512 |
| Dimension of vectors in full connection layer | dim_fc | 256 |
| Adam Optimizer's $\beta_1$ | beta1 | 0.9 |
| Adam Optimizer's $\beta_2$ | beta2 | 0.98 |
| Adam Optimizer's $\epsilon$ | epsilon | 10e−9 |
| Number of iterations to train all samples | epochs | 100 |

### 4.3 Comparison with Other Methods

We compare our method with some other methods, including: (1) machine learning method: AppScanner [23], this method extracted flows from the original encrypted traffic, then extracted 54 statistical features from individual flows, and finally classifies them using a machine learning classifier, which is described to be the best for random forests. FlowPrint [24] extracted features from network flows, clustered them according to network destinations, and finally constructed fingerprints based on the strong correlation between destinations. (2) Deep learning baseline method: 1D-CNN is a 1D convolutional neural network based encrypted flow method proposed by [5] which first applies end-to-end methods to the field of encrypted flow classification. Deep Fingerprinting [25] considered only the direction of the packets and then uses CNN for classification. CNN-LSTM is a method proposed by [7], which took three consecutive packets in a flow as model input, and extracted the features of the packets using CNN and LSTM for classification. Also, to illustrate the effectiveness of our proposed cuflow for classification, we compare the cases of using packet and flow, respectively.

**Table 3.** Comparison results of three classification tasks on three types of datasets.

| Experiment | A | | | | B | | | | C | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Datasets | ISCX-Tor [20] | | | | ISCX-VPN [21] | | | | USTC-TFC [21] | | | |
| Evaluation | AC | PR | RC | F1 | AC | PR | RC | F1 | AC | PR | RC | F1 |
| Appscanner [23] | 0.6722 | 0.4453 | 0.4598 | 0.4367 | 0.8211 | 0.7332 | 0.7225 | 0.7197 | 0.7814 | 0.7784 | 0.7791 | 0.7604 |
| FlowPrint [24] | 0.8917 | 0.6409 | 0.6019 | 0.6314 | 0.8830 | 0.6704 | 0.6599 | 0.6786 | 0.8443 | 0.7792 | 0.7893 | 0.7614 |
| 1D-CNN [5] | 0.9390 | 0.8809 | 0.9010 | 0.8791 | 0.9052 | 0.8614 | 0.8635 | 0.8513 | 0.8993 | 0.8756 | 0.8704 | 0.8630 |
| DF [25] | 0.7736 | 0.7689 | 0.7610 | 0.7602 | 0.7118 | 0.7136 | 0.7112 | 0.7011 | 0.7791 | 0.7427 | 0.7412 | 0.7367 |
| CNN-LSTM [7] | 0.9488 | 0.9241 | 0.9011 | 0.9246 | 0.9299 | 0.9253 | 0.9130 | 0.8832 | 0.9606 | 0.9673 | 0.9674 | 0.9613 |
| CUFT (flow) | 0.9298 | 0.9004 | 0.9376 | 0.9124 | 0.9003 | 0.8660 | 0.8121 | 0.8422 | 0.9812 | 0.9810 | 0.9791 | 0.9831 |
| CUFT (packet) | 0.9442 | 0.9172 | 0.9061 | 0.9134 | 0.9644 | 0.9678 | 0.9673 | 0.9598 | 0.9884 | **0.9899** | 0.9804 | 0.9823 |
| CUFT (cuflow) | **0.9967** | **0.9970** | **0.9978** | **0.9974** | **0.9833** | **0.9767** | **0.9737** | **0.9750** | **0.9918** | 0.9880 | **0.9908** | **0.9892** |

### 4.4 Results Analysis

According to the experimental results in Table 3, the deep learning approach is significantly better than the machine learning approach in all three tasks. Our method performs well in the three classification scenarios corresponding to the three datasets, and all outperform the other classification methods. The comparison results with packet-level and flow-level show that although both of them are effective for classification, our proposed cuflow is more effective for classification. Compared with the packet-level, our cuflow-level does not focus only on the inline relationships between the payloads of individual packets, but also on the contextual relationships between packets. Compared with flow-level, packets in cuflow are more correlated and connected, and thus the classification of extracted features is better. The result proves the effectiveness of the CUFT method. According to the results of the group cryptanalysis theory [26], it is

known that different cryptographic implementations have different degrees of randomness and the payload of encrypted traffic is pseudo-random, thus the CUFT method is feasible.

**Experiment A.** Since Tor's encrypted traffic results from three layers of encryption, the classification method based on statistical features and fingerprint construction does not represent the encrypted traffic well in such classification scenarios with the extracted features and thus performs poorly. Comparing our method with the other deep learning methods, our method achieves an F1 value of 99.74% in this classification task, which is 11.83% better than the 1D-CNN model, 23.72% better than Deep Fingerprinting and 7.28% better than the CNN-LSTM model. Table 4 shows the precision, recall and F1 of each category recognition are excellent in this task, which shows that our approach can well capture the spatiotemporal characteristics of Tor traffic from the ISCX-Tor dataset and can accurately classify 16 classes of applications with mixed Tor and regular encrypted traffic.

**Experiment B.** This classification task is slightly worse than the other two classification tasks, but still better than the other classification methods. To figure out the details of the error, we plot the confusion matrix, as shown in Fig. 6. We find that Streaming-type encrypted traffic has a 5.41% probability of being misclassified as VoIP, while VoIP has a 2.85% probability of being misclassified as Streaming, either as a result of our inability to label the dataset accurately or because audio and video are somewhat similar. Email is the category with the lowest classification accuracy. The reason may be that Email contains other elements such as text, which has some similarity with other categories, and secondly, there are fewer samples of this category in the dataset. The model does not learn the characteristics of Email type traffic very well.


**Fig. 6.** The mixing matrix of Experiment B.

**Experiment C.** We observe that each classifier performs well, which may be due to the fact that the application layer of malicious traffic in this dataset contains some unencrypted data [19], which allows the classifier to use this plaintext information to improve the classification accuracy. As shown in Table 4, the difference in metric values between categories is large, and some of the categories can reach a metric value of 1, but the accuracy of Zeus is only 90.91%. We analyzed and found that the training samples of Zeus were significantly smaller than other classes when model training was performed, and the model failed to fully learn its characteristics, resulting in slightly poorer classification results.

**Table 4.** Detailed experimental results for the three classification tasks.

| Experiment | Label | PR | RC | F1 | Label | PR | RC | F1 |
|---|---|---|---|---|---|---|---|---|
| A | Audio | 1.0000 | 1.0000 | 1.0000 | Tor:Audio | 1.0000 | 1.0000 | 1.0000 |
| | Browsing | 0.9916 | 0.9875 | 0.9896 | Tor:Browsing | 1.0000 | 0.9881 | 0.9940 |
| | Chat | 0.9965 | 0.9931 | 0.9948 | Tor:Chat | 1.0000 | 0.9964 | 0.9981 |
| | FT | 1.0000 | 1.0000 | 1.0000 | Tor:FT | 1.0000 | 1.0000 | 1.0000 |
| | Email | 1.0000 | 1.0000 | 1.0000 | Tor:Email | 1.0000 | 1.0000 | 1.0000 |
| | P2P | 0.9748 | 1.0000 | 0.9872 | Tor:P2P | 1.0000 | 1.0000 | 1.0000 |
| | Video | 0.9894 | 1.0000 | 0.9947 | Tor:Video | 1.0000 | 1.0000 | 1.0000 |
| | VoIP | 1.0000 | 1.0000 | 1.0000 | Tor:VoIP | 1.0000 | 1.0000 | 1.0000 |
| B | Chat | 0.9701 | 0.9882 | 0.9790 | VPN:Chat | 0.9901 | 1.0000 | 0.9950 |
| | Email | 0.9731 | 0.9187 | 0.9452 | VPN:Email | 0.9697 | 0.9912 | 0.9803 |
| | FT | 0.9841 | 0.9677 | 0.9758 | VPN:FT | 0.9760 | 1.0000 | 0.9878 |
| | Streaming | 0.9732 | 0.9416 | 0.9571 | VPN:Streaming | 1.0000 | 0.9802 | 0.9900 |
| | VoIP | 0.9311 | 0.9735 | 0.9518 | VPN:VoIP | 1.0000 | 0.9756 | 0.9877 |
| C | BitTrt | 1.0000 | 1.0000 | 1.0000 | Cridex | 1.0000 | 0.9865 | 0.9932 |
| | Facetime | 1.0000 | 1.0000 | 1.0000 | Geodo | 1.0000 | 1.0000 | 1.0000 |
| | FTP | 1.0000 | 1.0000 | 1.0000 | Htbot | 0.9934 | 0.9868 | 0.9901 |
| | Gmail | 0.9792 | 0.9792 | 0.9792 | Miuref | 0.9636 | 1.0000 | 0.9815 |
| | MySQL | 1.0000 | 1.0000 | 1.0000 | Neris | 0.9677 | 0.9677 | 0.9677 |
| | Outlook | 1.0000 | 1.0000 | 1.0000 | Nsis | 1.0000 | 0.9924 | 0.9962 |
| | Skype | 1.0000 | 1.0000 | 1.0000 | Shifu | 1.0000 | 1.0000 | 1.0000 |
| | SMB | 0.9853 | 1.0000 | 0.9925 | Tinba | 1.0000 | 1.0000 | 1.0000 |
| | Weibo | 1.0000 | 1.0000 | 1.0000 | Virut | 0.9615 | 0.9259 | 0.9433 |
| | Wow | 1.0000 | 0.9773 | 0.9885 | Zeus | 0.9091 | 1.0000 | 0.9524 |

## 5 Conclusion and Future Work

In this paper, we propose a new network encryption traffic classification method, CUFT, divided into two parts: data pre-processing and classification. To verify the effectiveness and generality of CUFT, we set up three types of complex traffic

classification tasks on three publicly available datasets. The experiments show that CUFT outperforms other machine and deep learning classification methods in all three types of classification tasks. In future work, we will focus on two areas:

(1) In traffic analysis, the labels of real-world encrypted traffic are difficult to re-tagged manually, and most of the traffic is unmarked. In the future, we will use semi-supervised learning or unsupervised learning to achieve classification in a limited number of labeled samples.
(2) In order to improve the quality of network services, real-time classification of encrypted traffic is also necessary, and we will consider the classification of not only offline but also online to accommodate the demand for real-time network traffic classification.

**Acknowledgements.** This work is supported by National Natural Science Foundation of China under Grants 62172307, U21A20466, 62272350, and 62272238.

## References

1. Wang, Y., Zhou, H., Feng, H., Ye, M., Ke, W.: Network traffic classification method basing on CNN. J. commun. **39**(1), 14–23 (2018)
2. Montieri, A., Ciuonzo, D., Aceto, G., Pescapé, A.: Anonymity services Tor, I2P, JonDonym: classifying in the dark (web). IEEE Trans. Dependable Secure Comput. **17**(3), 662–675 (2018)
3. Rezaei, S., Liu, X.: Deep learning for encrypted traffic classification: an overview. IEEE Commun. Mag. **57**(5), 76–81 (2019)
4. Hu, X., Gu, C., Wei, F.: CLD-Net: a network combining CNN and LSTM for internet encrypted traffic classification. Secur. Commun. Netw. **2021**, 1–15 (2021)
5. Wang, W., Zhu, M., Wang, J., Zeng, X., Yang, Z.: End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In: 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 43–48. IEEE (2017)
6. Wang, W., Zhu, M., Zeng, X., Ye, X., Sheng, Y.: Malware traffic classification using convolutional neural network for representation learning. In: 2017 International Conference on Information Networking (ICOIN), pp. 712–717. IEEE (2017)
7. Zou, Z., Ge, J., Zheng, H., Wu, Y., Han, C., Yao, Z.: Encrypted traffic classification with a convolutional long short-term memory neural network. In: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 329–334. IEEE (2018)
8. Cheng, J., et al.: MATEC: a lightweight neural network for online encrypted traffic classification. Comput. Netw. **199**, 108472 (2021)
9. Shapira, T., Shavitt, Y.: FlowPic: encrypted internet traffic classification is as easy as image recognition. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 680–687. IEEE (2019)
10. Zhao, R., et al.: An efficient intrusion detection method based on dynamic autoencoder. IEEE Wirel. Commun. Lett. **10**(8), 1707–1711 (2021)

11. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
12. 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. **24**(3), 1999–2012 (2020)
13. Al-Naami, K., et al.: Adaptive encrypted traffic fingerprinting with bi-directional dependence. In: Proceedings of the 32nd Annual Conference on Computer Security Applications, pp. 177–188 (2016)
14. Anderson, B., McGrew, D.: Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1723–1732 (2017)
15. Williams, N., Zander, S., Armitage, G.: A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. ACM SIGCOMM Comput. Commun. Rev. **36**(5), 5–16 (2006)
16. 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. **16**, 2367–2380 (2021)
17. Zhao, R., et al.: A novel traffic classifier with attention mechanism for industrial internet of things. IEEE Trans. Industr. Inform. (2023)
18. 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. **190**, 107974 (2021)
19. 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, pp. 633–642 (2022)
20. Lashkari, A.H., Draper-Gil, G., Mamun, M.S.I., Ghorbani, A.A., et al.: Characterization of tor traffic using time based features. In: ICISSp, pp. 253–262 (2017)
21. 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), pp. 407–414 (2016)
22. 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. **116**, 58–73 (2017)
23. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: AppScanner: automatic fingerprinting of smartphone apps from encrypted network traffic. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 439–454. IEEE (2016)
24. Van Ede, T., et al.: FlowPrint: semi-supervised mobile-app fingerprinting on encrypted network traffic. In: Network and Distributed System Security Symposium (NDSS), vol. 27 (2020)
25. 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, pp. 1928–1943 (2018)
26. Hu, X., Zhao, Y.: Block ciphers classification based on random forest. In: Journal of Physics: Conference Series, vol. 1168, p. 032015. IOP Publishing (2019)