ATVITSC: A Novel Encrypted Traffic Classification Method Based on Deep Learning

Ya Liu, Xiao Wang, Bo Qu, and Fengyu Zhao

***Abstract*— The increasing prevalence of encrypted communication on the modern internet has presented new challenges for traffic classification and network management. Traditional traffic classification methods cannot handle encrypted traffic effectively. Meanwhile, many existing methods either rely on hand-crafted features or fail to extract the underlying interaction patterns between data packets adequately. In this paper, we propose a novel encrypted traffic classification method called the Attention-based Vision Transformer and Spatiotemporal for Traffic Classification (ATVITSC). In the preprocessing stage, packet-level images within a session, generated from the payload of data packets, are combined into a session image to mitigate information confusion. In the classification stage, session images are first processed by the packet vision transformer (PVT) module, which employs the transformer encoder and multi-head self-attention mechanism, to capture the global features. In parallel, session images are also processed by the spatiotemporal feature extraction (STFE) module, where spatial features of packets are extracted by the convolution operation with the attention mechanism and temporal features between packets are then combined by the bidirectional Long Short-Term Memory (LSTM). The global and spatiotemporal features are fused in the feature fusion classification (FFC) module by a dynamic weighting mechanism and encrypted traffic is finally classified based on the fused features. Comprehensive experiments on various types of encrypted traffic, including virtual private network (VPN), onion router (Tor), malicious traffic, and mobile traffic, show that the ATVITSC successfully improves the macro-f1 scores to 97.88%, 98.79%, 99.67%, 94.90%, respectively. The results also reveal that the ATVITSC exhibits better classification performance and generalization ability than the state-of-the-art methods.**

***Index Terms*— Encrypted traffic classification, self-attention mechanism, a dynamic weighting mechanism, spatialtemporal network.**

Received 19 December 2023; revised 9 June 2024; accepted 16 July 2024. Date of publication 25 July 2024; date of current version 10 October 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62002184, in part by the Doctoral Startup Fund of Guangdong University of Science and Technology under Grant XJ2023002001, in part by the Open Topics from The Lion Rock Labs of Cyberspace Security under Project #LRL24017, and in part by the Scientific Research Initiation Fund for High-Level Talents of Shanghai Publishing and Printing College under Grant 2024rcky22. The associate editor coordinating the review of this article and approving it for publication was Prof. Ghassan Karame. *(Corresponding author: Bo Qu.)*

Ya Liu is with the School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China, and also with the Lion Rock Labs of Cyberspace Security, CTIHE, Hong Kong, China.

Xiao Wang is with the School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China.

Bo Qu is with the School of Computer Science, Guangdong University of Science and Technology, Dongguan 523083, China (e-mail: bo@qubo.im).

Fengyu Zhao is with the Department of Information and Intelligence Engineering, Shanghai Publishing and Printing College, Shanghai 200093, China.

Digital Object Identifier 10.1109/TIFS.2024.3433446

I. Introduction

In RECENT years, network security has been elevated to a significant strategic position in many countries. Encryption technology is one of the most widely used methods to safeguard network security [1]. However, cybercriminals can also be hidden in the encrypted network traffic. To address this challenge, encrypted traffic classification has been extensively applied in intrusion detection systems and malware propagation detection. It seems impossible to classify the encrypted traffic if the encrypted algorithm is applied safely. Nevertheless, this hypothesis is not always correct. Lin et al. [2] evaluated the randomness of 5 ciphers through 15 sets of statistical tests, and found that all these ciphers could not achieve the ideal randomness. Furthermore, because encrypted communication increases the payload and latency of the network, different types of encrypted traffic may result in different impacts on the network. Hence, if encrypted traffic is classified correctly, network administrators can better understand and manage various communication behaviors [3].

Early traffic classification relies on port numbers, where different types of network traffic can be classified based on their source and destination ports [4]. However, with the widespread usage of dynamic and non-standard ports, port-based methods become less reliable. Deep Packet Inspection (DPI) methods [2], which classify network traffic by analyzing the content of network packets, are more precise than port-based methods but only applicable to unencrypted traffic and require substantial computational resources. Machine learning is also used to classify network traffic by analyzing the behavior or statistical features of network traffic, such as the transmission frequency, size, and flow duration of packet [5]. Although this method is more robust than port-based and DPI methods, the performance depends on manually designed features.

The application of deep learning in traffic classification can automatically extract features, which avoids the laborious steps of feature engineering [6]. Deep learning models can adapt to different encryption scenarios and perform end-to-end encrypted traffic classification [7]. Encrypted traffic classification based on deep learning has been a hot spot of network security in recent years. However, on the one hand, previous research studies extracted the spatial or temporary features of network flows and the relationships among packets inadequately, resulting in the limited ability of feature extraction and insufficient robustness of classification [8]. On the other hand, some methods that attempt to capture the spatiotemporal features of traffic may lead to confusion in temporal informa-

1556-6021 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

tion. For example, in the cascaded structure of convolutional neural networks and recurrent neural networks, convolutional operations are based on local neighborhoods, which may lead to a lack of temporal relationships among the spatial features extracted from packets, thereby affecting the extraction of spatiotemporal features [9].

To tackle the aforementioned problems, we propose a novel scheme called Attention-based Vision Transformer and Spatiotemporal for Traffic Classification (ATVITSC). In the preprocessing stage, the raw traffic is split into network sessions. Within each session, the packet sequence is transformed into a sequence of packet images, which are combined into a session image. All session images are divided into fixed-size patches to avoid information distortion. In the classification stage, session images are fed into the packet vision transformer (PVT) and spatiotemporal feature extraction (STFE) modules in parallel to capture the global and spatiotemporal features. In the PVT module, the patches are transformed through packet embedding, positional embedding, and length embedding, which are added together to feed into a PVT encoder with a Multi-Head Self-Attention (MHSA) mechanism to extract global features of the network flow. In the STFE module, the patches are transformed through the attention-based convolution operation and the bidirectional LSTM network to capture spatial features of each packet within the session and temporal features among packets, thereby obtaining spatial-temporal features of the entire network session. The global and spatial-temporal features are fused in the feature fusion classification (FFC) module by a dynamic weighting mechanism and encrypted traffic is finally classified based on the fused features. The main contributions of this paper are summarized as follows:

(1) Propose a packet-level preprocessing method to transform encrypted traffic into session images, which not only mitigates the issue of information disorder but also enhances the capability to learn inner relations among packets. In addition, this preprocess method adopts both position encoding and length encoding to capture the order and length information among packets, which alleviates the problem of information loss after image generation.

(2) Design the PVT and STFE modules with the attention mechanism, which accurately identifies network traffic. In the PVT module, three kinds of embedding operations are designed, and then added together to feed into a PVT encoder with the MHSA mechanism to retain the time and position information and extract global features fully. In the STFE module, the convolution operation with the self-attention mechanism and bidirectional LSTM network are designed so that ATVITSC can accurately capture spatiotemporal features.

(3) Propose a dynamic weighting mechanism that adaptively adjusts the weights of global and spatial-temporal features, thereby resulting in the appropriate fusion of these two features. Additionally, by introducing a temperature parameter, ATVITSC avoids the highly imbalanced weight distribution of different features during the feature fusion process. Meanwhile, it ensures that different feature extraction modules perform sufficient learning during the training stage.

(4) A large number of comprehensive experiments on four datasets, i.e., USTC-TFS, ISCX-Tor, ISCX-VPN and Cross-Platform (U.S. Android APP), are performed to demonstrate the superior performances of ATVITSC. From the view of the accuracy, Macro-F1, Macro-Recall and Macro-Precision, ATVITSC outperforms ten advanced schemes in most cases. In addition, we also explain the reasons for selecting hyper-parameters by conducting experiments, such as the number of chosen packets, the first m bytes selected from each packet and the temperature.

The rest of the paper is structured as follows: Section II introduces the related work on encrypted traffic classification. Section III presents our novel approach ATVITSC. Section IV provides a detailed experimental analysis of ATVITSC. Finally, section V concludes the paper.

II. Related Work

The current approaches employed for traffic identification can be categorized into port-based methods, payload-based methods, machine learning methods that rely on statistical features and end-to-end deep learning methods [10].

As discussed above, port-based traffic classification and payload-based methods such as DPI cannot be suitable for analyzing the encrypted traffic. Machine learning methods use statistical features to perform encrypted traffic classification. For instance, AppScanner [11] trained a random forest classifier using statistical features based on the packet size, while BIND [12] used statistical features based on time features. However, these features are only suitable for specific scenarios, and it is difficult to design a universal set of statistical features that can be adapted to different encryption scenarios.

In recent years, encrypted traffic classification methods based on deep learning have developed very rapidly. Liu et al. proposed an FS-Net based on recurrent neural networks (RNNs) to automatically extract the simplistic features from the sequence of packet sizes for the encrypted traffic classification [13]. Shapira and Shavitt used FlowPic [14] to extract packet size and arrival time from raw traffic to construct images for classification. Wang et al. used 1D-CNN [15] and 2D-CNN [16] approaches for the traffic classification by converting the first 768 bytes of network flows into grayscale images. However, they didn't handle them at the packet level, thereby resulting in the confusion of packet image information from different periods during the convolutional operations. Shen et al. proposed GraphDApp based on graph neural networks to classify encrypted traffic by constructing a Traffic Interaction Graph (TIG) [17]. However, compared to other methods, GraphDApp takes more time to label unknown flows and the accuracy decreases when adjustments are made to the application fingerprints. Huoh et al. [18] considered the raw bytes, metadata and inter-packet relationships in traffic to construct a traffic graph. They used graph neural networks to extract graph representations for classification. In traffic recognition, Wang [19] stressed the significance of detecting essential byte features, and analyzed the impact of the first 25 bytes, the first 100 bytes and the least helpful 300 bytes of the payload for encrypted traffic classification to indicate the initial 200 bytes were the most crucial. Lotfollahi et al.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

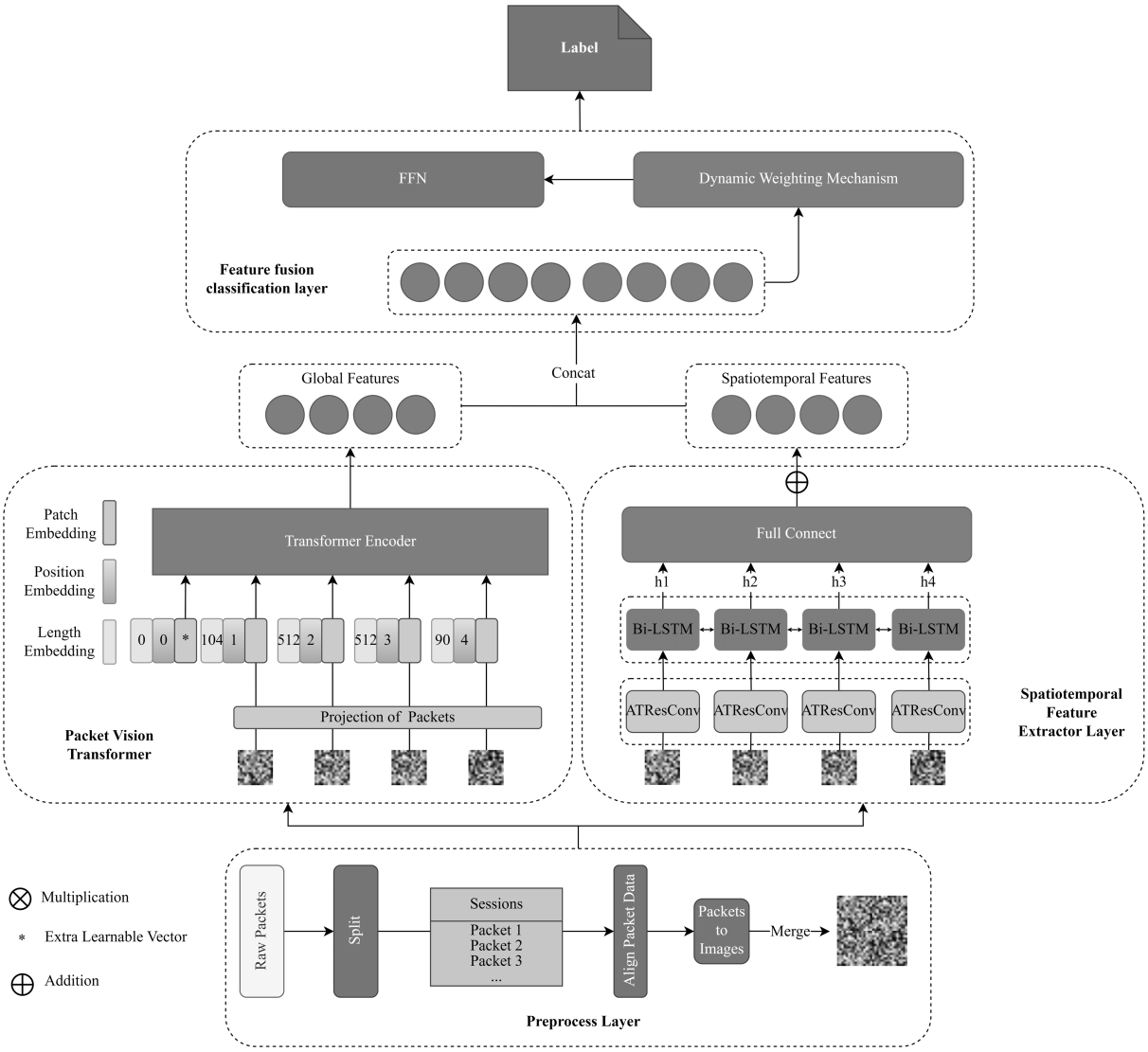


Fig. 1. The frame of ATVITSC.

discovered that 96% of packet payloads have a length of less than 1480 bytes [20]. Hence, they used the first 1500 bytes as the model's input and considered protocol fields in the network and transport layers. Taking the entire payload as input will increase the computational complexity of the model. However, these fields are primarily designed for network transmission rather than application classification. Thus, protocol fields below the application layer usually contain minimal helpful information. Excessive irrelevant information would increase the data complexity and potentially lead the model to rely on unimportant details, thus reducing its ability to differentiate fine-grained traffic classification. Yao et al. proposed HAN [21] by combining the attention mechanism with LSTN to extract temporal features from raw traffic. Kumano et al. [22] demonstrated that the minimum number of packets needed for processing certain features was 10. This result provided a basis for reducing the number of packets required. Lin et al. proposed TSCRNN [23] by combining CNN and LSTM in a cascade structure, which only extracted temporal and spatial features from network flows and did not fully capture the relationships among packets in the flow. Therefore, TSCRNN cannot extract the global features of the entire flow. The CMTSNN [24] proposed by Zhu et al. also used a cascaded structure of CNN and LSTM. In a word, these methods do not adequately capture the dependencies among local features and the interaction among packets of the network flow.

III. ATVITSC

This section introduces a novel method ATVITSC, shown in Figure 1. It comprises a preprocessing layer, the packet vision transformer, the spatiotemporal feature extraction layer, and the feature fusion classification layer. First, the raw traffic is partitioned into bidirectional flows called sessions. Second, extract a portion of the payload of each data packet within one session to generate a byte-level grayscale image for each packet, which is then combined to construct session-level grayscale images. Next, these session-level grayscale images are input into the PVT and STFE modules to capture global and spatiotemporal features, respectively. Finally, a dynamic

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

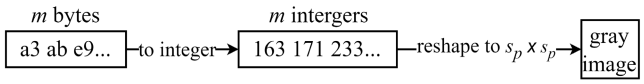


Fig. 2. Generation of grayscale image for application layer data of the packet.

weighting mechanism (DW) is proposed in the feature fusion classification layer to perform the feature fusion, thereby accurately achieving the encrypted traffic classification.

A. Raw Packets Preprocess

The raw traffic, including various types of packets, is divided into flows to achieve a more precise classification. IP addresses, port numbers, and protocol types are typically used to identify and organize flows [25]. Specifically, the original network traffic data can be denoted by R as follows: $R = \{(t_i, s_i, sp_i, d_i, dp_i)\}_{i=1}^{n}$, where t_i, s_i, d_i, sp_i and dp_i represent the timestamp of the i-th packet, the source IP address, the destination IP address, the source port and destination port number, respectively. Partition R into different network flows: $S = \{S_1, S_2, \ldots, S_m\}$. Each flow S_j contains a set of packets with the same source IP address s_j, the same source port sp_j, the same destination IP address d_j and the same destination port dp_j: $S_j = \{(t_{j,k}, s_{j,k}, sp_{j,k}, d_{j,k}, dp_{j,k})\}_{k=1}^{n_j}$, where n_j represents the number of packets in the j-th flow.

In the ATVITSC scheme, we mainly focus on a bidirectional flow called a session. The source IP, the destination IP, the source port, and the destination port in a session can be interchangeable. Therefore, the session contains more interactive information than a unidirectional flow. According to the timestamps of the bidirectional flows in chronological order, combine the bidirectional flows to generate a session flow $S_{session}$, i.e., $S_{session} = S_{Src \to Dst} \cup S_{Dst \to Src}$, where $S_{Src \to Dst}$ represents the flow of packets with the source IP as Src, the destination IP as Dst and the source port as SrcPort, and $S_{Dst \to Src}$ represents the flow of packets with the source IP as Dst, the destination IP as Src and the source port as SrcPort. The symbol $\cup$ denotes the combined operation. During the combining process, the packets are arranged by the timestamps in chronological order.

After extracting the sessions, remove those packets irrelevant to encrypted traffic classification to eliminate the dataset's noise and redundancy. The protocol fields below the application layer contain almost no useful information. They are mainly used for network transmission rather than network recognition. Additionally, IP addresses should not be used for classification learning because they may cause the model to focus on packet sources too much. Therefore, only the application layer's data packets of each session flow should be retained. Fixed-step sampling is used for data augmentation, i.e., a fixed number of packets are sampled in the session to increase the dataset's richness and the model's robustness [26].

At the final stage of preprocessing, ATVITSC proposes a novel method of constructing packet-level session images, effectively alleviating the confusion of packet information. The process of converting the payload of each packet into a grayscale image is shown in Figure 2. Precisely, for a given session flow $S_{Session}$, we extract the first m bytes of the application layer's data for each of the n packets, where m is a perfect square ${s_p}^2$ and n is also a perfect square. The application layer's data will be truncated if its length is more than m bytes. Otherwise, it will be padded with 0×00. Similarly, the number of packets will be padded to n if the number of packets is less than n. Read each packet byte by byte, convert each byte to an integer, and transform each integer to a packet image of size $s_p \times s_p$, represented as $P_i \in \mathbb{R}^{s_p \times s_p}$. Finally, the packet images are combined into a session image. The combined operation is shown in Figure 3 for the case of $n = 4$ and $m = 256$.

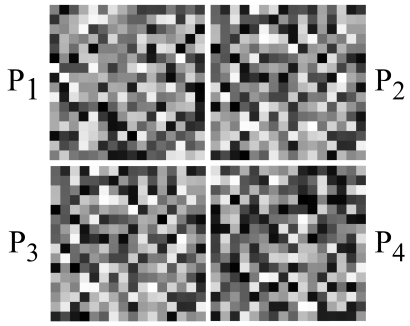


Fig. 3. The combined operation of the packet image, P_i represents the image of the i-th packet in the session flow.

B. Packet Vision Transformer

This subsection proposes a global feature extraction method called packet vision transformer (PVT), which is inspired by vision transformer (ViT) [27]. PVT contains a session embedding layer and applies a multi-head self-attention mechanism. In a session embedding layer, PVT decomposes session images into a sequence of multiple packet images to perform embedding processing. Then, using the multi-head self-attention mechanism, PVT captures the interaction information among all packets and extracts global features from the session flows.

1) Embedding Operations:

a) Packet embedding: During the embedding process, the session image is divided into multiple packet images, and each is projected into a d-dimensional space. Specifically, we perform a two-dimensional convolution operation on the session image. The convolution operation has a stride of s_p, a kernel size of $s_p \times s_p$, and a channel size of d. Subsequently, we obtain the embedding $e_i \in \mathbb{R}^d$ for each packet.

b) Position embedding: Due to the use of self-attention mechanisms, PVT cannot capture the sequential information directly. However, since the packets in the session are chronological, we can embed the position of each packet to preserve their orders and position information. In the ATVITSC scheme, PVT combines sine and cosine functions to obtain sequential details on data packets and better capture their intrinsic dependencies. Specifically, let $p_i \in \mathbb{R}^d$ represent the position embedding of the i-th data packet in the session, which can be expressed as:

$$p_{i,2j} = \sin\left(\frac{\text{pos}_i}{10000^{2j/d}}\right), \quad p_{i,2j+1} = \cos\left(\frac{\text{pos}_i}{10000^{2j/d}}\right)$$

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

where j represents the j-th dimension of the positional embedding and pos_i denotes the position of the i-th data packet in the session flow. This formula is derived from the absolute positional embeddings used in Transformers [28].

c) Length embedding: After converting the packets into images, the information on the length of packets is lost. Introducing the length embeddings for the packets could mitigate this deficiency. Since the length of each packet is a discrete value, we convert it into a one-hot vector and then generate a length embedding by using a linear layer. Let len_i denote the length of the i-th packet, which can be transformed into a one-hot vector $lv_i \in \mathbb{R}^{\text{max\_len}}$ as follows:

$$lv_{i,j} = \begin{cases} 1, & j = \text{len}_i \\ 0, & j \neq \text{len}_i \end{cases}$$

By using learnable parameter $W_{\text{len}} \in \mathbb{R}^{\text{d}\times\text{max\_len}}$, we can compute the length embedding $l_i \in \mathbb{R}^{\text{d}}$ by $l_i = W_{\text{len}} lv_i$.

Let max_len be 1500 due to the Ethernet standard's maximum transmission unit (MTU) of the data link layer frame being set to 1500 bytes [29]. Finally, find the sum of the packet embeddings e_i, position embeddings p_i and length embeddings l_i to obtain the final embeddings $h_i \in \mathbb{R}^d$ for each packet: $h_i = e_i + p_i + l_i, i = 1, 2, \ldots, n$.

Similarly, inspired by BERT [30], a learnable embedding $V_s \in \mathbb{R}^d$ is added at the beginning of these embeddings to represent the global features of the session. The output after feeding the session into the embedding layer is computed by $E_s = [V_s; h_1; h_2; \ldots; h_n]$, where $[a; b]$ denotes the new vector obtained by concatenating vectors a and b along the time dimension.

2) Packet Vision Transformer: The encoder of PVT uses the multi-head self-attention mechanism to capture global dependencies and process features through a feed-forward neural network, enabling efficient modeling and feature extraction of packet image sequences. The encoder of PVT, as shown in Figure 4, follows a similar structure to the Transformer's encoder. The k-head self-attention mechanism projects queries, keys, and values into k subspaces based on the self-attention mechanism and performs separate self-attention operations in each subspace. Specifically, for the i-th head, the query matrix(Q_i), key matrix(K_i) and value matrix(V_i) are calculated by the linear projection as follows:

$$Q_i = E_s W_i^Q, K_i = E_s W_i^K, V_i = E_s W_i^V$$

where $W_i^Q \in \mathbb{R}^{d\times d_h}$, $W_i^K \in \mathbb{R}^{d\times d_h}$ and $W_i^V \in \mathbb{R}^{d\times d_v}$ are learned weight matrices, d_h is typically set to d/k.

Next, we use the scaled dot-product attention mechanism [31] to compute the self-attention weight matrix:

$$A_i = softmax(\frac{Q_i K_i^T}{\sqrt{d_h}}), \quad SA_i = A_i V_i$$

where A_i denotes the attention weight matrix for the i-th head, $\sqrt{d_h}$ represents the scaling factor used to solve the problem of gradient explosion, and SA_i denotes the output of the self-attention mechanism of the i-th head.

After concatenating the outputs of all attention heads, we obtain the final output of the multi-head self-attention layer by applying a linear projection: $MHSA(E_s) = [SA_1, SA_2, \ldots, SA_k]W_{mhsa}$, where k represents the number of heads, $[\cdot]$ denotes concatenation along the last dimension and $W_{mhsa} \in \mathbb{R}^{k\cdot d_h \times d}$ is a learned weight matrix.

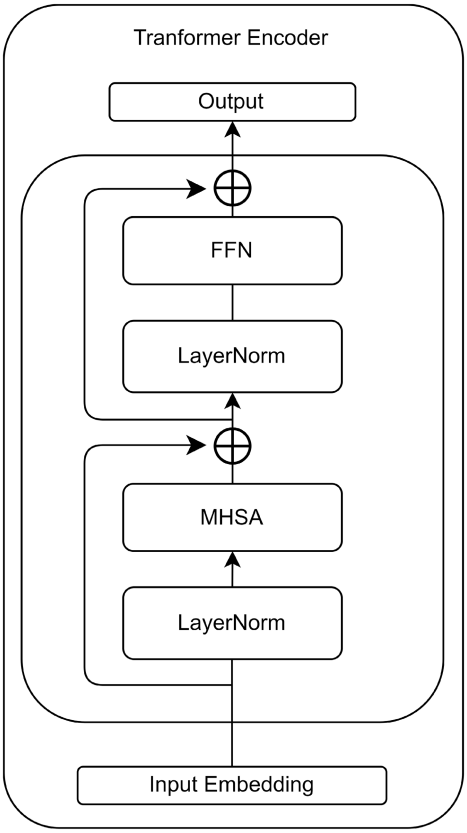


Fig. 4. Packet Vision Transformer encoder.

Algorithm 1 demonstrates the process of the packet vision transformer. The PVT encoder consists of multiple submodules, each of which includes a multi-head self-attention mechanism (MHSA) and a fully connected layer (FC) with a LeakyRELU [32] activation function. Layer normalization (LN) [33] and residual connections [34] are utilized in each sub-layer as well.

This architecture design enables the encoder to conduct multi-level transformations and generate representations of input sequences, thereby enhancing the ability of PVT to capture the semantic and contextual information embedded in the input sequences. It can be formalized as:

$$O'^{(l_{\text{layer}})} = \text{MHSA}(LN(E_s{}^{(l_{\text{layer}})})) + E_s{}^{(l_{\text{layer}}-1)}$$
$$O^{(l_{\text{layer}})} = \text{FC}(LN(O'^{(l_{\text{layer}})})) + O'^{(l_{\text{layer}})}$$

where L is the number of layers of the PVT encoder, and $l_{\text{layer}} \in [0, L]$ is the l_{layer}-th layer.

To obtain this global feature, we extract the first vector from the output sequence O^L of the final layer in the PVT encoder, which is regarded as the global feature of the session: $F_{global} = O_0^L$.

C. Spatiotemporal Feature Extractor Module

The spatiotemporal feature extraction(STFE) module consists of two core components: the residual Attention

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 The Workflow of PVT

Input: the normalized network session image S_n, the session length sequence L_n, the global feature dimension d_{model}, the number of transformer encoder n_{layers}.

Output: The global feature O_{global};

PKE = PaketEmbedding(S_n,Conv2d(s_p,s_p));
LE = LengthEmbedding(L_n,Embedding(1500,d_{model})
PE = PositionEmbedding(S_n)
IE = Addition(PKE,LE,PE)
for $i = 1$ to n_{layers} **do**
    LN_{ie} = LayerNorm(IE);
    W_{mhsa} = MultiHeadSelfAttention(LN_{ie},d_{model});
    RC= Addition(W_{mhsa},IE)
    IE = FeedForward(RC,d_{model})
end for
F_{global} = IE
return F_{global};

Algorithm 2 The Workflow of STFE

Input: Normalized network session image S_n, Temporal feature dimension d_t, Spatial feature dimension d_s, The number of AttetionConvResNet n_c.

Output: The spatiotemporal feature F_{st};

Split S_n into a sequence of packet images P_s;
$T = Length(P_s)$
t = 1
for X in P_s **do**
    F_m = BN(Conv2d(X));
    $W_{spatial}$ = ConvAttention(GAP(F_m),GMP(F_m),d_s);
    $F_{spatial}$ = MatMul(F_m,$W_{spatial}$);
    F'_m = Addition(Shortcut($F_{spatial}$),F_m);
    $F_{s(t)}$ =Flatten(Conv$_{1\times1}$(F'_m));
    $t = t + 1$;
end for
h_f, h_b = BiLSTM($F_{s(1)}$, $F_{s(2)}$, $F_{s(3)}$, ..., $F_{s(T)}$);
F_{st} = FC(Concat($h_{f,n}$, $h_{b,n}$),d_s);
return F_{st};

Convolutional layer (ResAtConv) with residual connections and the attention mechanism and the Bidirectional LSTM layer (Bi-LSTM). ResAtConv extracts high-level spatial features of each packet and solves the problem of temporal information disorder while convolving the entire session image. Then, these spatial features with temporal features are fed into Bi-LSTM [35] to capture the spatiotemporal features of a session further. Algorithm 2 shows the process of the spatiotemporal feature extraction module.

1) Residual Attention Convolution Layer: Inspired by SeNet [36], we propose a Residual Attention Convolutional Module. This module primarily consists of a convolutional layer and an attention layer. Its structure is shown in Figure 5. The convolutional layer is responsible for the feature extraction of the packet images. Because the contribution of each channel's feature map is different, the ATVITSC scheme presents an attention layer to determine the weights of each channel.

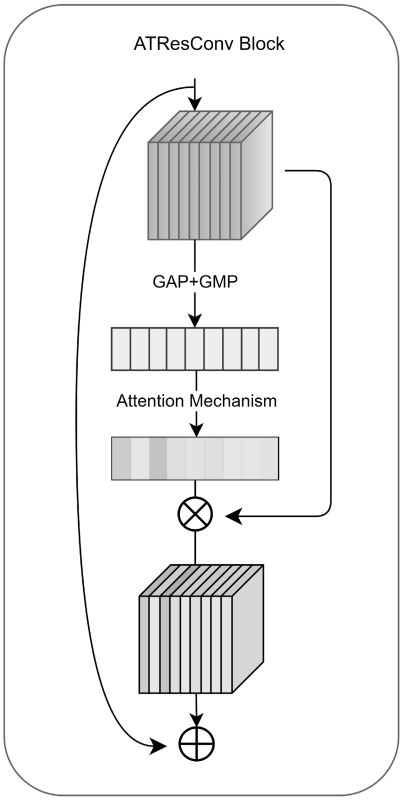


Fig. 5. Residual Attention Convolution.

The session image is segmented into packet images, and convolutional operations are performed on each packet to generate a set of feature maps F_m. By applying global max pooling (GMP) and global average pooling (GAP) on F_m, each feature map is compressed into a scalar [37] which still retains the important information of F_m, and F_m will be compressed into two vectors (the dimension is equal to the number of channels of F_m) which will be fed to the fully connected (FC) layer with an S-shaped Activation function. Next, the feature map is weighted by using the attention values of each channel. This weight will focus on feature maps with more significant contributions while suppressing unimportant feature maps. Finally, a 1×1 convolutional kernel is applied to adjust the features at different positions. Similar to PVT, we also apply batch normalization (BN) [38] and residual connections [34] in this module. This process can be formalized as:

$$\begin{cases} F_m = BN(Conv(X)) \\ F'_m = FC(FC(GAP(F_m)) + FC(GMP(F_m)))F_m + F_m \\ F''_m = Conv_{1\times1}(F'_m) \end{cases}$$

Due to the small size of individual data packet images, multiple ResAtConv operations may lead to overfitting. Thus ATVITSC only uses one ResAtConv operation for each packet image, which can learn the weights of each channel, allowing the model to focus on more critical feature maps during the convolution process. Finally, 1×1 convolution is used to adjust the information of different positional features in the weighted feature map to highlight critical spatial features. The output of

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

the last layer of ResAtConv is flattened to obtain the spatial features F_s of a packet: $F_s = Flatten(F''_m)$.

2) Bidirectional LSTM: After obtaining spatial features of each packet through ResAtConv, these features are then sequentially fed into the Bi-LSTM. The nature of Bi-LSTM allows it to simultaneously capture the forward and backward information of the packet sequence, which can enhance the modeling ability of ATVITSC to capture the correlations among packets. Each packet's spatial features $F_{s(t)}$ is fed into the LSTM cell in both temporal order and reverse temporal order to capture the temporal dependencies of the spatial features $F_{s(t)}$.

$$h_{f,t} = \text{LSTM}(F_{s(t)}, h_{f,t-1}), \quad h_{b,t} = \text{LSTM}(F_{s(t)}, h_{b,t+1})$$

where $h_{f,t}$ represents the forward hidden state of the LSTM at time step t and $h_{b,t}$ represents the backward hidden state of the LSTM at time step t. Concatenate the last hidden states from both the forward and backward hidden state sequences to obtain the spatiotemporal features of the session. Subsequently, these concatenated hidden states undergo a nonlinear transformation by using a fully connected layer (FC) with the LeakyRELU activation function: $F_{st} = FC([h_{f,n}, h_{b,n}])$, where n represents the maximum value of the time step, i.e., the length of the packet sequence.

D. Feature Fusion Classification Layer

Traditional methods, such as simple concatenation or fixed weighted fusion, cannot fully utilize the correlations between these two features and cannot adaptively adjust the weights. In the ATVITSC scheme, we propose a dynamic weighting mechanism in the Feature Fusion Classification layer (FFC) to compute the weight coefficients of the global feature F_{global} and the spatiotemporal feature F_{st}. After the weighted fusion of F_{global} and F_{st}, the fused feature is fed into two fully connected layers (FC) and the softmax function to generate the classification probability distribution. The Algorithm 3 describes the workflow of FFC.

Algorithm 3 The Workflow of FFC

Input: Global feature of session F_{global}, Spatiotemporal feature of session F_{st}.
Output: Result of classification;

$F_{g,s}$ = Concat(F_{global},F_{st});
α_g, α_s = DynamicWeight($F_{g,s}$);
v = Matmul&Addition(α_g, α_s, F_{global}, F_{st});
y = Softmax(FeedForwad(v));
$ClassResult$ = ArgMax(y);
return $ClassResult$;

The dynamic weighting mechanism assigns different weights for F_{global} and F_{st} of different sessions. Moreover, we also introduce a temperature parameter τ to control the balance of weights among different feature extraction modules and to prevent the feature extraction module from failing during the training process [39]. It can be formalized as:

$$z = \tanh(W_{g,s} \cdot [F_{global}; F_{st}] + b_{g,s}), \quad \xi = W_z z + b_z$$

$$\alpha_g = \frac{\exp(\xi_1/\tau)}{\exp(\xi_1/\tau) + \exp(\xi_2/\tau)}, \alpha_s = \frac{\exp(\xi_2/\tau)}{\exp(\xi_1/\tau) + \exp(\xi_2/\tau)}$$

where $W_{g,s}$, $b_{g,s}$, W_z and b_z are learnable parameters, $\tanh(\cdot)$ represents the hyperbolic tangent function, $\exp(\cdot)$ represents the exponential function, $\xi \in \mathbb{R}^2$ is a unnormalized weight for the feature, and $\xi_1 \in \mathbb{R}^2$ and $\xi_2 \in \mathbb{R}^2$ are separate unnormalized weights for the global and local features. The temperature parameter τ is a hyperparameter that balances the attention of FFC to different features and alleviates extreme cases of weight coefficients when normalizing them. How to select τ will be discussed in section IV.

After computing the normalized weights for different features, the fusion vector v can be represented as: $v = \alpha_g \cdot F_{global} + \alpha_s \cdot F_{st}$. Finally, v is fed into two fully connected layers, and a non-linear transformation is applied to the hidden layer to obtain a higher-level representation. The softmax function, i.e., $y = \text{softmax}(\text{FC}(\text{LeakyRELU}(\text{FC}(v))))$, is used to determine the category of the encrypted traffic.

IV. Experimental

In this section, we evaluate the performance of ATVITSC on four different scenarios of encrypted traffic: Virtual Private Networks (VPN), Tor (the Onion Router), malicious traffic, and Android mobile traffic. The implementation configurations of the experiments are shown in Table I. Among these parameters, ATVITSC selects a standard batch size and learning rate in deep learning classification tasks to strike a balance between its training speed and its performance. Additionally, it employs the cross-entropy as the loss function due to the cross-entropy demonstrating the efficacy of previous classification tasks, which effectively quantifies the disparity between model predictions and the truth labels [40]. The selection of the Adam optimizer is due to its favorable convergence properties during the training process, adaptive learning rate adjustments, and ability to handle sparse gradients [41]. PyTorch is chosen as the deep learning framework to support the model's design and training processes in our experiments. We compare ATVITSC with ten state-of-the-art traffic classification methods and further investigate the roles of its sub-modules through an ablation study. These experiments aim to demonstrate the generalization and effectiveness of ATVITSC in various encryption environments.

A. Datasets

We select four widely used datasets to study the performance of ATVITSC. These datasets are listed in the following:

- USTC-TFS2016 dataset [16]: This dataset contains samples of encrypted network traffic from 10 categories of benign traffic and ten categories of malicious traffic.
- Tor-nonTor dataset (ISCXTor2016) [42]: This dataset is designed to classify encrypted applications on Tor, which contains samples from 16 different applications.
- VPN-nonVPN dataset (ISCXVPN2016) [43]: This dataset focuses on encrypted traffic classification on virtual private networks (VPNs), which comprises both VPN and non-VPN traffic samples from 6 communication applications.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

TABLE I
EXPERIMENTAL CONFIGURATION

Hyperparameter	Value
Packet Numbers	16
Bytes	256
Training set allocation	70%
Verification set allocation	10%
Test set allocation	20%
Batch size	16
Warm-up rate	0.1
Dropout rate	0.5
Loss function	Cross Entropy
Optimizer	Adam
Number of epochs	30
Learning rate schedule	StepLR
Learning rate	5×10^{-4}
Packet Vision Transformer depth	2 layers
Multi-head attention mechanism head count	12
Attention convolutional residual layer count	2
Feedforward neural network hidden layer size	2048
Global feature dimensions	256
Spatiotemporal feature dimensions	256
Temperature parameter	500
Deep learning Framework	Pytorch version 1.12.0
CUDA version	11.6

- Cross Platform (US Mobile APP) [44]: This dataset provides a large amount of mobile traffic for analysis and evaluation, which contains 59 Android traffic from the United States for specific classification tasks.

These datasets provide abundant samples and features, allowing us to verify the validation of ATVITSC and evaluate its performance in real-world scenarios. By studying and analyzing these datasets, we enhance the understanding of encrypted traffic classification in various encryption environments.

B. Metrics

To evaluate the performance of the ATVITSC, we use four metrics: Accuracy, Macro-F1, Macro-Recall, and Macro-Precision. Accuracy is a standard metric in classification schemes, representing the ratio of correctly classified samples to total samples [45]. Macro-F1, Macro-Recall, and Macro-Precision are evaluation metrics designed for multi-classification problems [46]. These metrics can be expressed using the following formula:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}, Recall = \frac{TP}{TP + FN}$$
$$Precision = \frac{TP}{TP + FP}, \quad F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

where TP, TN, FP, and FN represent true positive, true negative, false positive, and false negative.

$$F1_{macro} = \frac{1}{N_c} \sum_{i=1}^{N_c} F1_i, \quad Rc_{macro} = \frac{1}{N_c} \sum_{i=1}^{N_c} Recall_i$$
$$Pr_{macro} = \frac{1}{N_c} \sum_{i=1}^{N_c} Precision_i$$

where N_c represents the number of classes, $F1_i, Recall_i$ and $Precision_i$ are the F1-score, the Recall score and the Precision score for class i.

C. Exploration of Hyperparameters

We explore three important hyperparameters: the number of chosen data packets (denoted by Packet Number), the first m bytes selected from each data packet (denoted by ByteNum), and the temperature coefficient of ATVITSC in the ISCX VPN dataset to determine how to select the appropriate hyperparameters.

1) Packet Number and ByteNum: We do a series of experiments to explain the selections of two parameters, i.e., Packet Number and ByteNum. First, to select the appropriate number of data packets, we consult previous research, especially Kumano's results, which show that selecting 10 data packets during the visualization process can achieve relatively good classification results [22]. To maximize the search space, we choose the number of data packets to be 4, 9, 16, 25, 36, and 49. Second, since Wang have demonstrated that the traffic after 300 bytes of the payload has little significance for improving the traffic classification [19], we choose ByteNum being 16, 36, 64, 100, 169, 256, 324, and 400. For all combinations of selected parameters, the accuracy of the ATVITSC in the test set is shown in Figure 6. The results show that the performance of ATVITSC tends to stabilize when the Packet Number is more than 9, and the accuracy of ATVITSC is high when ByteNum is between 100 and 300 bytes. Finally, after careful analysis, the accuracy of ATVITSC reaches its optimal value of 0.9436 when the Packet Number is 16 and ByteNum is 256.

2) Temperature Parameter: We study the selection of the temperature parameter τ in the feature fusion classification layer (FFC). This parameter τ depends on a dynamic weighting mechanism that adjusts the weight distribution between global and spatiotemporal features to achieve adaptive fusion of different features. Randomly select 500 samples from each category in four different datasets for training. During the training process, τ is set from 1 to 800 in step 20. After 20 rounds of training, Figure 7 shows the change of accuracy under different values of τ. The accuracy of ATVITSC significantly improves at larger values of τ. Conversely, when τ is set to a small value, the weight distribution between global features and spatiotemporal features may not be reasonable, resulting in a decrease in the performance of ATVITSC. In particular, when τ is more than 400, the classification accuracy tends to stabilize. By reasonably selecting the value of τ, we can effectively improve the performance of the ATVITSC and avoid unreasonable weight distribution during the initial stage of training. In a word, the temperature parameter τ, which plays a crucial role in the FFC module, is of great significance for achieving feature fusion and improving the performance of ATVITSC.

D. Comparison With Other Methods

We compare ATVITSC with other state-of-the-art traffic classification schemes on the USTC-TFS, ISCX-Tor,

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

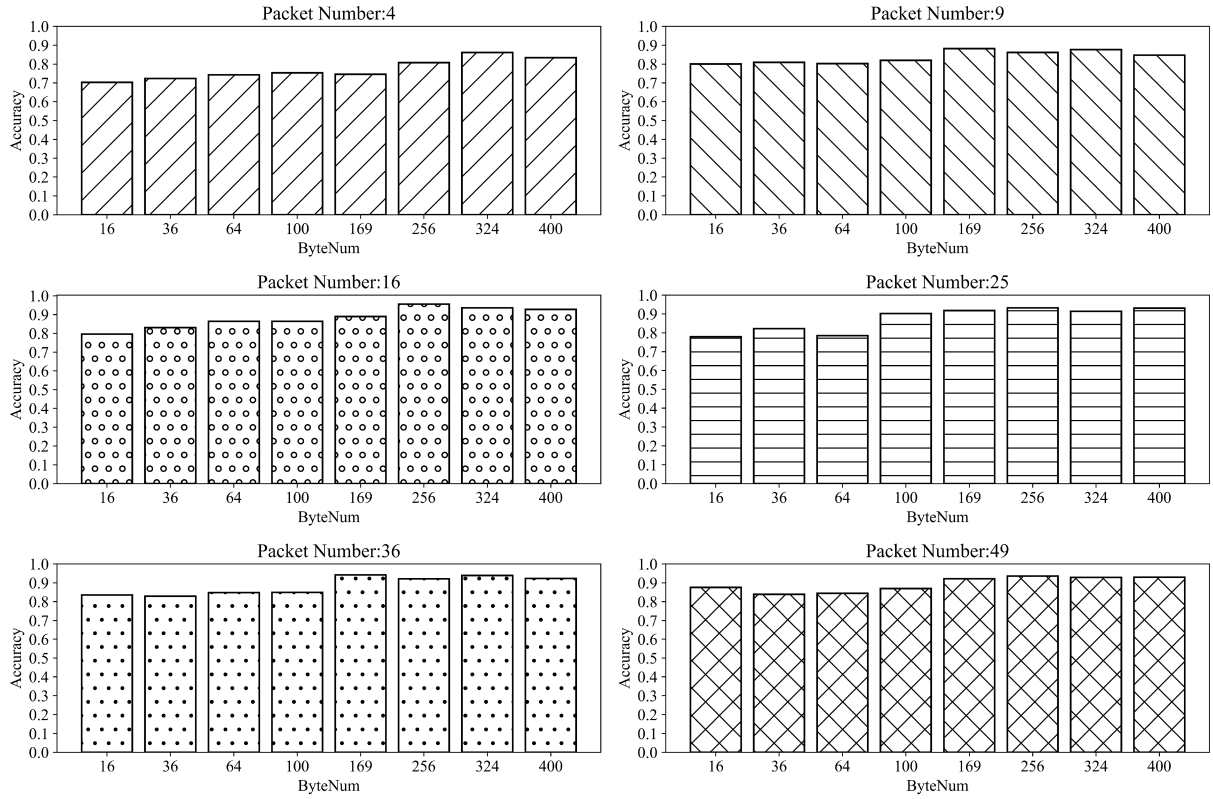


Fig. 6. Accuracy under different combinations of the number of data packets and the first m bytes of each packet.

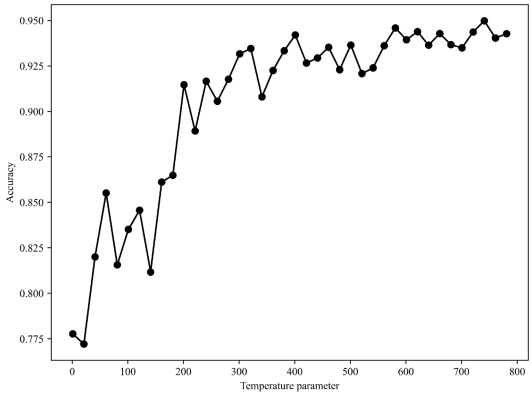


Fig. 7. Accuracy variation curves under different values of τ.

ISCX-VPN and Cross-Platform (US Android APP) datasets, including some classic schemes based on machine learning such as AppScanner [11] and BIND [12], and some advanced schemes based on deep learning such as FS-Net [13], GraphDApp [17], FlowPic [14], HAN [21], TSCRNN [23] and DeepPacket [20], CMTSNN [24], Flow-GNN [18]. These schemes based on deep learning cover almost all neural network structures. For example, FS-Net used a structure with recurrent neural networks to classify temporal features, FlowPic converted statistical features into images and used convolutional neural networks for classification, GraphDApp used graph neural network structure for encrypted traffic classification, TSCRNN and CMTSNN combined recurrent and convolutional neural network architecture, DeepPacket used autoencoder and convolutional neural network structure, HAN combined the attention mechanism and recurrent neural network for encrypted traffic classification. Flow-GNN uses geometric deep learning that simultaneously considers raw bytes of packets, metadata features, and the relations among packets for classification. These schemes cover various popular neural network structures and algorithms in the current field of deep learning, which makes us comprehensively compare the performances and effectiveness of different advanced encrypted traffic classification schemes based on deep learning.

1) Malicious Traffic Classification: According to the experimental results provided in Table II, ATVITSC achieves the best result in terms of Rc_{macro} and $F1_{macro}$. However, four other methods, such as DeepPacket, TSCRNN, CMTSNN, and Flow-GNN, also show considerable performance on the UTSC-TFC dataset because the UTSC-TFC dataset contains plaintexts in the application layer of malicious traffic. These plaintexts make it easier for the models to learn patterns of different categories from unencrypted data and achieve accurate classification. Table III shows the experimental results on a 20-classification task of USTC-TFC.

2) Tor Traffic Classification: In Table IV, ATVITSC achieves an accuracy of 98.79%, which is 3.79% higher than the previously best result, TSCRNN, because TSCRNN fails to fully capture the relationships among packets and only extracts spatiotemporal features by using a cascaded structure. In contrast, ATVITSC designs the Packet Vision Transformer to extract global features and a dynamic weighting mechanism

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

TABLE II
COMPARISON RESULTS ON USTC-TFC

Method	Pr_{macro}	Rc_{macro}	$F1_{macro}$	Accuracy
AppScanner	89.84%	89.68%	88.92%	89.54%
BIND	86.81%	83.82%	83.96%	84.57%
FS-Net	88.46%	89.20%	88.40%	88.46%
GraphDApp	82.26%	82.60%	82.34%	87.89%
FlowPic	97.21%	97.29%	97.34%	97.44%
HAN	96.97%	96.23%	96.28%	96.31%
TSCRNN	98.70%	98.60%	98.70%	98.68%
DeepPacket	96.50%	96.31%	96.41%	96.40%
CMTSNN	98.76%	98.84%	98.81%	98.55%
Flow-GNN	**99.70**%	99.59%	99.61%	**99.74**%
ATVITSC	99.66%	**99.67**%	**99.67**%	99.66%

TABLE III
CLASSIFICATION REPORT OF USTC-TFC

Type	Precision	Recall	F1-score
BitTorrent	99.81%	99.73%	99.77%
Cridex	99.99%	100.00%	99.99%
FTP	99.99%	99.80%	99.90%
Facetime	99.84%	100.00%	99.92%
Geodo	99.84%	100.00%	99.92%
Gmail	99.11%	99.34%	99.23%
Htbot	100.00%	99.62%	99.81%
Miuref	100.00%	99.99%	99.99%
MySQL	99.41%	99.95%	99.68%
Neris	98.46%	100.00%	99.22%
Nsis-ay	99.98%	99.81%	99.89%
Outlook	98.84%	98.55%	98.69%
SMB	99.91%	98.93%	99.41%
Shifu	99.89%	99.99%	99.94%
Skype	98.92%	100.00%	99.46%
Tinba	100.00%	99.94%	99.97%
Virut	99.95%	98.42%	99.18%
Weibo	99.69%	99.63%	99.66%
WorldOfWarcraft	99.88%	99.82%	99.85%
Zeus	99.87%	100.00%	99.94%

TABLE IV
COMPARISON RESULTS ON ISCX TOR

Method	Pr_{macro}	Rc_{macro}	$F1_{macro}$	Accuracy
AppScanner	37.56%	44.22%	39.13%	67.22%
BIND	45.98%	45.15%	45.11%	71.85%
FS-Net	50.80%	53.50%	45.90%	60.71%
GraphDApp	48.64%	48.23%	44.88%	68.36%
FlowPic	86.93%	84.23%	85.74%	90.49%
HAN	87.57%	90.66%	88.26%	91.42%
TSCRNN	94.90%	94.80%	94.80%	95.00%
DeepPacket	75.49%	73.99%	74.73%	74.49%
CMTSNN	93.79%	94.63%	93.34%	93.26%
Flow-GNN	94.82%	95.77%	95.30%	94.88%
ATVITSC	**98.80**%	**98.79**%	**98.79**%	**98.79**%

to fuse global and spatiotemporal features for classification. Meanwhile, the accuracy of ATVITSC also surpasses DeepPacket's by 24.30%, because DeepPacket has a weaker generalization ability. Table V shows the experimental results of a 16-classification task on Tor2016.

3) VPN Traffic Classification: Table VI demonstrates all experimental results on the dataset ISCX VPN. We find that the accuracy of ATVITSC is about 97.89%, which is more than the accuracy of Flow-GNN, CMTSNN, DeepPacket, and TSCRNN by 0.33%, 4.59%, 4.6%, and 6.19%, respectively. Although Flow-GNN can achieve similar accuracy,

TABLE V
CLASSIFICATION REPORT OF ISCX TOR

Type	Precision	Recall	F1-score
AUDIO	96.49%	97.55%	97.02%
BROWSING	94.93%	97.35%	96.12%
CHAT	97.02%	95.95%	96.48%
FILE	98.64%	98.15%	98.40%
MAIL	99.85%	99.55%	99.70%
P2P	97.88%	99.10%	98.48%
TorAUDIO	99.70%	99.75%	99.72%
TorBROWSING	99.95%	99.65%	99.80%
TorCHAT	100.00%	100.00%	100.00%
TorFILE	99.90%	100.00%	99.95%
TorMAIL	100.00%	100.00%	100.00%
TorP2P	99.75%	99.80%	99.78%
TorVIDEO	99.90%	99.95%	99.92%
TorVOIP	99.95%	100.00%	99.98%
VIDEO	97.99%	95.25%	96.60%
VOIP	98.90%	98.70%	98.80%

TABLE VI
COMPARISON RESULTS ON ISCX VPN

Method	Pr_{macro}	Rc_{macro}	$F1_{macro}$	Accuracy
AppScanner	73.99%	72.25%	71.97%	71.82%
BIND	75.83%	74.88%	74.20%	75.34%
FS-Net	75.02%	72.38%	71.31%	72.05%
GraphDApp	60.45%	62.20%	60.36%	59.77%
FlowPic	91.86%	91.48%	91.55%	92.03%
HAN	89.12%	88.62%	88.37%	89.71%
TSCRNN	92.70%	92.60%	92.60%	91.70%
DeepPacket	93.77%	93.06%	93.21%	93.29%
CMTSNN	94.10%	91.60%	92.80%	93.30%
Flow-GNN	93.55%	96.27%	94.83%	97.56%
ATVITSC	**97.89**%	**97.89**%	**97.88**%	**97.89**%

TABLE VII
CLASSIFICATION REPORT OF ISCX VPN

Type	Precision	Recall	F1-score
Chat	97.64%	99.25%	98.44%
File	97.94%	97.40%	97.67%
Mail	99.35%	100.00%	99.68%
P2P	100.00%	100.00%	100.00%
Streaming	89.49%	93.25%	91.33%
VPNChat	99.45%	99.55%	99.50%
VPNFile	99.50%	99.00%	99.25%
VPNMail	100.00%	99.85%	99.92%
VPNP2P	99.95%	99.90%	99.92%
VPNStreaming	99.45%	99.85%	99.65%
VPNVoip	99.60%	99.90%	99.75%
Voip	92.39%	86.80%	89.51%

the $F1_{macro}$ of ATVITSC is more than Flow-GNN's by 3.05%. In fact, ISCX VPN is an imbalanced dataset. In this dataset, TSCRNN uses random sampling to enhance data diversity, DeepPacket uses undersampling to mitigate the data imbalance, and CMTSNN, designed only for encrypted traffic classification in IoT, adopts the cost penalty matrix and improved cross-entropy loss function without the attention mechanism. In contrast, ATVITSC uses the attention mechanism for each module to explore the interactive relationship among packets in ISCX VPN fully. Please refer to Table VII for more detailed experimental results on twelve classification tasks performed on VPN2016.

4) Mobile Traffic Classification: We also perform ATVITSC and other ten schemes on Cross-Platform (US

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

TABLE VIII
COMPARISON RESULTS ON CROSS-PLATFORM

Method	Pr_{macro}	Rc_{macro}	$F1_{macro}$	Accuracy
AppScanner	24.41%	24.23%	24.15%	39.71%
BIND	32.44%	31.85%	30.28%	48.56%
FS-Net	35.65%	35.57%	35.36%	51.21%
GraphDApp	30.76%	32.79%	31.39%	44.78%
FlowPic	85.34%	86.94%	85.45%	87.49%
HAN	88.33%	83.39%	87.85%	90.48%
TSCRNN	92.10%	88.84%	86.70%	91.43%
DeepPacket	82.84%	76.27%	80.71%	86.44%
CMTSNN	90.34%	88.12%	87.72%	90.85%
Flow-GNN	91.98%	92.53%	91.38%	92.86%
ATVITSC	**94.91**%	**95.24**%	**94.90**%	**96.21**%

Android APP). Table VIII gives comparison results. From Table VIII, ATVITSC demonstrates superior performance across four key metrics again. Specifically, its $F1_{macro}$ score surpasses those of HAN, CMTSNN and Flow-GNN by 7.05%, 7.18% and 3.52%, respectively. Although HAN also uses attention mechanisms, it has certain limitations in extracting temporal features. CMTSNN only extracts features at the spatiotemporal dimension level. Flow-GNN achieves suboptimal classification results by utilizing statistical features and constructing a graph of the order among the original data packets. In contrast, ATVITSC not only captures the interactions among data packets but also extracts spatiotemporal features from network flows, resulting in more accurate classification. The classification report for 59 categories of U.S. Android app traffic in the Cross-platform is listed in Table IX.

5) Comparison of Inference Speed: We list the inference time of 9 different deep learning methods in Table X when the batch size is 16. Since AppScanner and BIND are based on machine learning, we do not analyze them. First, FS-Net has the fastest inference speed of 18.67ms because FS-Net only accepts simple sequence features as input. However, simple sequence features also affect its classification accuracy. Second, FlowPic uses a simple CNN structure, but its input matrix (1500×1500) is too large, resulting in its inference time of 71.52ms. Although the model structure of ATVITSC is relatively complex, its input size is relatively small. For example, when Packet Number is 16 and ByteNum is 256 bytes, the size of input received by ATVITSC is only 80×80. So ATVITSC can achieve an inference speed of 32.54ms.

In summary, the inference speed of different models is influenced by the complexity of input features and model structure. FS-Net has advantages in speed, but its classification accuracy is relatively low. ATVITSC achieves a good balance between inference speed and classification accuracy.

6) Analysis and Discussion: According to the above experimental results, ATVITSC shows significant performance advantages over other state-of-the-art traffic classification methods on multiple datasets, while the performances of other methods are not stable enough on different datasets. The F1 value of TSCRNN in the USTC dataset is 98.70%, while it is 92.70% in the VPN dataset. The F1 value of DeepPacket in the USTC dataset is 96.50%, but it drops to 75.49% in the Tor dataset. AppScanner and Bind perform the worst in the three datasets, mainly due to the insufficient applicability of encrypted traffic classification methods based on statistical features in different datasets. It is necessary to redesign the features for each type of traffic. FS-Net uses packet sizes as features and then applies recurrent neural networks for computing. However, such features are too simplistic and make it difficult to accurately reflect the encrypted network traffic features. GraphDAPP constructs a TIG for encrypted traffic based on the packet length and direction. Then it applies a graph neural network for classification. However, the constructed graph structure may not capture the key features of encrypted traffic. Meanwhile, when the fingerprint of the application changes, the accuracy will correspondingly decrease. DeepPacket chooses the IP header and the first 1480 bytes of each packet to form a 1500-byte input, which contains the data below the application layer. However, this selection will result in excessive computation and data redundancy. In addition, the CNN model fails to effectively extract the relationship between temporal features and data packets, resulting in lower accuracy. FlowPic converts packet size and arrival time into images and uses CNN for classification. Using simplistic features as input makes it difficult to accurately extract traffic features. TSCRNN and CMTSNN combine CNN and LSTM in a cascaded structure to extract spatiotemporal features from network flows but do not consider the interaction among data packets. HAN combines LSTM with the attention mechanism, but it can only extract packet features from the temporal dimension, which is not comprehensive enough. Flow-GNN utilizes the sequential relationship among data packets, raw traffic and meta features as network inputs, which can fully extract traffic features for classification. However, due to its lack of attention mechanism, its ability to extract global features may be weak in certain scenarios.

ATVITSC achieves an F1 value of over 97% in datasets such as USTC, VPN, and Tor, and the F1 value on the Android Mobile traffic dataset also reaches 94.9%. Clearly, it demonstrates stable performance in various application environments. This advantage mainly stems from ATVITSC's comprehensive use of the PVT and STFE modules, augmented by various feature extraction methods, such as the attention mechanism, feedforward neural network, residual attention convolution module, and bidirectional LSTM. These methods collectively extract the spatiotemporal and spatial features of traffic data. In addition, the PVT module of ATVITSC first embeds each data packet in the traffic image to obtain the initial vector of the packet. It uses an attention mechanism to learn the interaction relationship among the packets and uses a feedforward neural network to extract the global features of the entire flow. Compared with traditional visualization methods, our proposed traffic visualization method can avoid the problem of temporal information distortion caused by convolution operations in network traffic.

After obtaining the spatial features of the data packet, bidirectional LSTM is used to accurately extract spatiotemporal features, and the dynamic weighting mechanism can dynamically adjust the weights of global and spatiotemporal features for feature fusion based on the features of the data. Compared to previous methods with fixed weights in advance,

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

TABLE IX
CLASSIFICATION REPORT OF CROSS-PLATFORM(U.S. ANDROID APP)

Type	Precision	Recall	F1-score	Type	Precision	Recall	F1-score
appinventor	100.00%	88.89%	94.12%	bibleverses	97.50%	78.00%	86.67%
vozdonarrador	100.00%	100.00%	100.00%	marcopolo	100.00%	94.44%	97.14%
ambatana	100.00%	95.00%	97.44%	adpmobile	93.94%	100.00%	96.88%
kindle	98.36%	100.00%	99.17%	mShop	96.32%	96.32%	96.32%
tahoe	97.06%	93.40%	95.19%	audible	100.00%	96.63%	98.29%
autopten	95.24%	93.02%	94.12%	babycenter	96.69%	98.65%	97.66%
carezone	92.96%	91.67%	92.31%	carmax	100.00%	100.00%	100.00%
learntodrawglow.c	73.21%	80.39%	76.64%	drawglowgomics	71.43%	60.61%	65.57%
contextlogic	100.00%	96.15%	98.04%	espn	100.00%	98.25%	99.12%
foxsports	100.00%	98.61%	99.30%	goodrx	94.74%	94.74%	94.74%
primer	84.62%	91.67%	88.00%	tachyon	100.00%	78.57%	88.00%
cardboarddemo	91.11%	85.42%	88.17%	iconology	100.00%	96.61%	98.28%
indeed	96.08%	94.23%	95.15%	jardogs	90.48%	100.00%	95.00%
musicplayer	100.00%	100.00%	100.00%	fastestflashlight	100.00%	90.00%	94.74%
jiubang	97.10%	97.10%	97.10%	matchmobile	100.00%	94.74%	97.30%
linewebtoon	100.00%	100.00%	100.00%	okcupid	90.20%	95.83%	92.93%
pandora	91.18%	96.88%	93.94%	newsbreak	96.63%	97.73%	97.18%
pinterest	100.00%	100.00%	100.00%	booster	100.00%	100.00%	100.00%
frontpage	87.10%	98.18%	92.31%	sephora	97.83%	94.74%	96.26%
fitness	90.79%	98.57%	94.52%	smartkeyboard	98.08%	98.08%	98.08%
spotify	98.81%	100.00%	99.40%	toutiao	96.93%	97.31%	97.12%
steam	97.30%	98.63%	97.96%	tekoia	98.11%	98.11%	98.11%
tool	94.44%	100.00%	97.14%	trulia	100.00%	100.00%	100.00%
ulta	100.00%	86.49%	92.75%	wSugarMommasDating	96.18%	96.18%	96.18%
xpro	93.55%	100.00%	96.67%	zillowmap	96.67%	97.75%	97.21%
zoosk	99.10%	98.21%	98.65%	hily	65.79%	96.15%	78.12%
epic	80.00%	100.00%	88.89%	chase.pawpatrol	98.51%	100.00%	99.25%
jojo.siwacall	91.80%	100.00%	95.73%	castbox	93.22%	90.16%	91.67%
zedge	95.65%	97.54%	96.59%	peel	98.15%	100.00%	99.07%
telepathic	86.96%	100.00%	93.02%				

TABLE X
CLASSIFICATION TIME FOR DIFFERENT MODELS WHEN BATCHSIZE IS 16

Model	Time(ms)	Model	Time(ms)
FS-Net	**18.67**	DeepPacket	21.84
GraphDApp	58.49	CMTSNN	28.09
FlowPic	71.52	Flow-GNN	125.36
HAN	323.25	TSCRNN	297.98
ATVITSC	**32.54**		

ATVITSC is more flexible and could enhance the applicability of the model in different environments. In contrast, other methods have limitations in feature extraction and model design, resulting in unstable or unsatisfactory performance. Meanwhile, they could not consider the interaction among data packets.

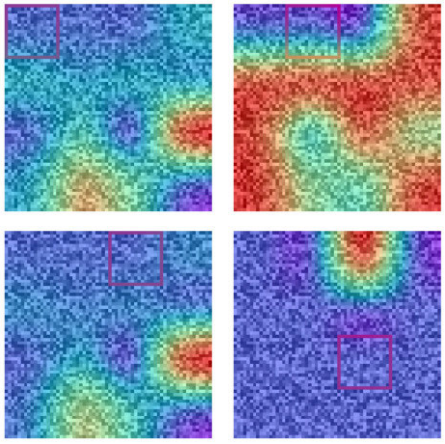


Fig. 8. Attention Visualization.

E. Interaction of Packets

In this experiment, let the number of packets be 16 and the payload size be 256 bytes while keeping other parameters unchanged. Our main objective is to observe the dependencies among packets by visualizing the attention in the PVT. To simplify the analysis, we train ATVITSC on the VPN dataset and randomly select a sample. The attention matrix computed by PVT is then mapped onto a session image. Figure 8 illustrates the attention distributions of different packets within the same session (for display purposes, only four packets are chosen). Different colors represent the attention level of the model towards different parts, with darker colors indicating greater attention. The red box represents a packet. Obviously, for different data packets, the model focuses on different parts, which further proves that ATVITSC can capture the interaction of different packets and extract global features adequately.

F. Dynamic Weighting Mechanism

We analyze the role of the dynamic weighting mechanism on the ISCX-Tor dataset. In the training process, without using the temperature parameter τ, significant fluctuations in the weight of the STFE and PVT modules lead to an unstable training process, as seen in Figure 9. After 400 steps of training, the model encounters an extreme weight distribution problem where the weights of the STFE approach 0. During the initial stage of training, if there is a significant difference in contribution between the two feature extraction modules, the

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

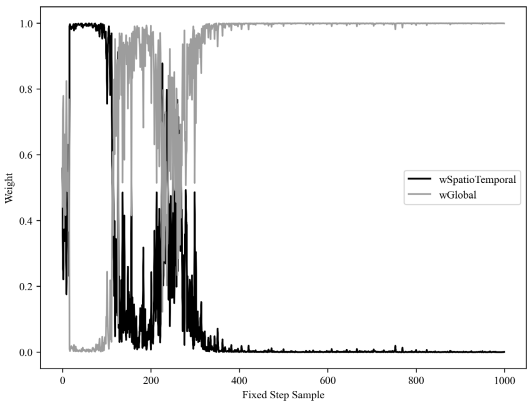


Fig. 9. Weight changes in training without τ.

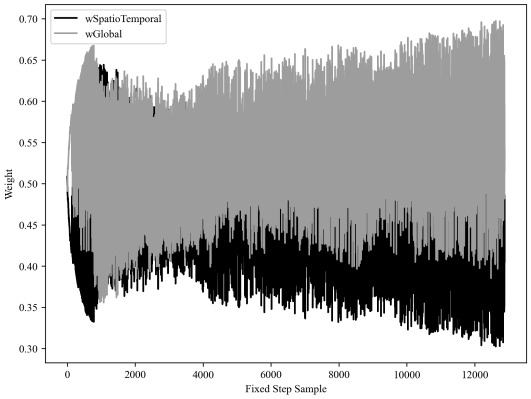


Fig. 10. Weight changes in training.

weight of one of the modules may approach 0. In this case, the weight of the PVT approaches 1, while the weight of the STFE module approaches 0. Consequently, the STFE is ignored after 400 steps, rendering it unable to continue learning. Therefore, without using the temperature parameter τ, the two sub-modules of ATVITSC cannot sufficiently complete the training.

With the temperature parameter τ, ATVITSC can achieve more balanced attention between the STFE and PVT modules. This adjustment makes both sub-modules effectively learn rather than directly disregard the weaker one. Thus ATVITSC allows the initially lower-weighted module to actively participate in feature extraction and classification processes. In our experiment, let τ be 500, which is a more reasonable change in the weights of the two modules. The weight changes of the two modules are depicted in Figure 10. Introducing τ alleviates the problem of extreme weight coefficient distributions. Both sub-modules effectively accomplish learning, and the fusion classification layer assigns corresponding weights to the global and spatiotemporal features of different sessions to complete traffic classification.

Compare the simple concatenation of global and spatiotemporal features with a dynamic weighting mechanism to those without a dynamic weighting mechanism. The accuracy variations of the two methods during the training process are shown in Figure 11. The results indicate that the dynamic

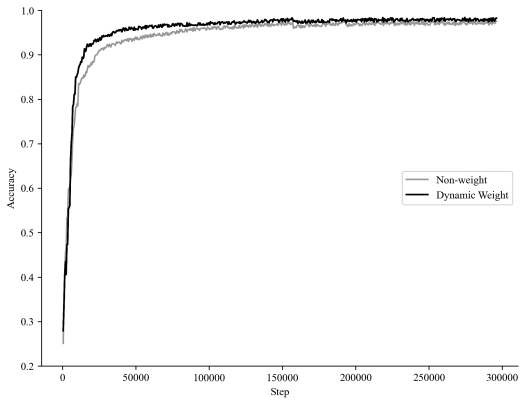


Fig. 11. Training accuracy on ISCX Tor with/without dynamic weighting mechanism.

TABLE XI
EVALUATION OF DIFFERENT PREPROCESS METHODS

Methods	Accuracy
Our Preprocess Method based on Packet-level+CNN	89.3%
Traditional Preprocess Method based on Session-level+CNN	82.7%

weighting mechanism can improve the convergence speed and accuracy of training. This result is attributed to the adaptive adjustment of the importance of different features during the training process, leading to a stable and effective model.

G. Comparison of Preprocess Methods

To validate the superiority of the proposed preprocess method in this paper, we compare it with a traditional approach of generating session-level images (using the first 1600 bytes of each session). Due to the incompatibility of the traditional method with the ATVITSC model, a widely used 1D-CNN model [15] is applied for performance comparison in this experiment. In the VPN dataset, 1000 sessions are randomly selected for each class. To ensure similar image sizes between the two methods, our preprocess approach uses the first 256 bytes of the payload and selects the first four packets of each session for image generation, which is compared with the traditional preprocess method based on session-level images. Table XI demonstrates that our method performs better. Generating images on the packet level allows the processed data to retain their temporal features and effectively prevent temporal information distortion from convolutional operations. Additionally, length embeddings are introduced to maintain the packets' length information. Conversely, simply extracting the first z bytes at the session level can lead to confusion and loss of information.

H. Ablation Study

Module ablation experiments and independent validation experiments need to be conducted on all modules to evaluate the performance of each module in the ATVITSC system. The ablation analysis of ATVITSC is performed on the ISCX-Tor dataset and the results are shown in Table XII. PVT exhibits the best performance, followed closely by STFE. Obviously,

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

TABLE XII
EVALUATION OF DIFFERENT MODULES

Modules	macro-accuracy	macro-precision	macro-recall	macro-f1
PVT	95.85 ± 1.23	95.51 ± 1.84	94.70 ± 1.05	94.96 ± 0.83
STFE	91.99 ± 2.46	91.74 ± 2.12	90.89 ± 1.91	90.99 ± 1.13
PVT+STFE	98.00 ± 0.66	97.58 ± 0.38	97.18 ± 0.57	97.35 ± 0.48
PVT+STFE+DW	98.79 ± 0.19	98.80 ± 0.39	98.79 ± 0.43	98.79 ± 0.32

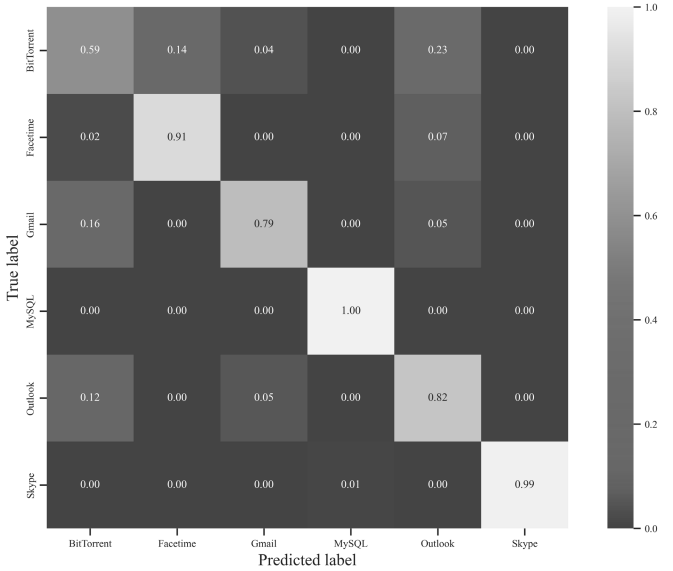


Fig. 12. Classification results of PVT in a single packet dataset.

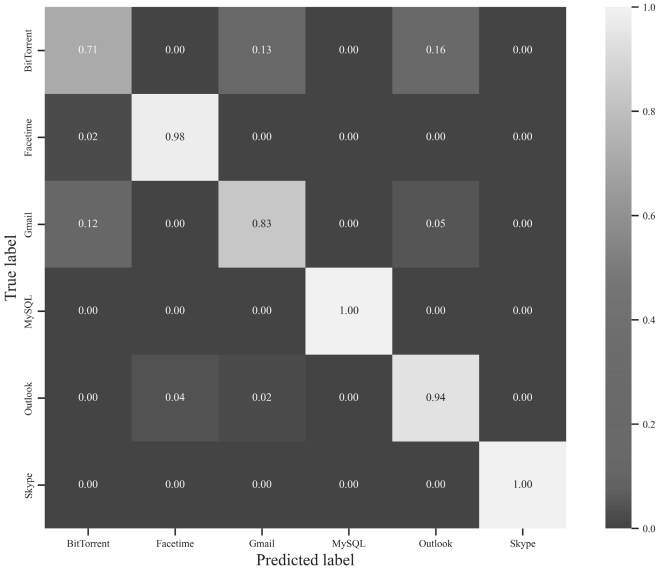


Fig. 13. Classification results of STFE in a single packet dataset.

the STFE's performance is relatively weaker than PVT's. When the PVT and STFE modules are combined, the network traffic features in both global and spatiotemporal dimensions can be considered comprehensively, leading to improved identification accuracy. Furthermore, more accurate classification performance is achieved by introducing the dynamic weighting mechanism to fuse the features extracted by the two modules. In a word, the dynamic weighting mechanism is better suited for combining different types of features.

We study the performances of PVT and STFE in sessions that only contain a single packet. A sample containing one packet is selected from six categories in the UTSC dataset: BitTorrent, Facetime, Gmail, MySQL, Outlook and Skype. The confusion matrices of the PVT and STFE are shown in Figure 12 and Figure 13. In this case, the accuracy of STFE is superior to that of PVT. With only one packet, there is no information to interact across locations, degrading the self-attention mechanism to a fully connected layer. Specifically, if the packet sequence contains only one packet, denoted as x_1, its embedding vector representation can be expressed as $x_1 \in \mathbb{R}^{1\times d}$, the query vector $Q \in \mathbb{R}^{1\times d_h}$, the key vector $K \in \mathbb{R}^{1\times d_h}$ and the value vector $V \in \mathbb{R}^{1\times d_v}$ in the following:

$$\begin{aligned} Q &= XW_Q = x_1W_Q, \\ K &= XW_K = x_1W_K, \\ V &= XW_V = x_1W_V \end{aligned}$$

where $W_Q \in R^{d\times d_h}, W_K \in R^{d\times d_h}, W_V \in R^{d\times d_v}$ are the weight matrices of Query, Key and Value, respectively, d_h and d_v represent the dimensions of Key and Value.

According to the calculation formula of the self-attention mechanism, the unnormalized score $S \in R^{1\times 1}$ is obtained by the values of Q and K:

$$S = \frac{QK^T}{\sqrt{d_h}} = \frac{x_1W_QW_K^Tx_1^T}{\sqrt{d_h}}$$

For a sequence $S = [S_1]$ that contains only one packet, the attention weights a can be obtained by the softmax function:

$$\begin{aligned} a = softmax\,(S)\,V &= \frac{\exp(s_1)}{\sum_{j=1}^{1}\exp(s_j)}(x_1W_V) \\ &= \frac{\exp(s_1)}{\exp(s_1)}(x_1W_V) = x_1W_V \end{aligned}$$

Based on the above formula, we can find that the self-attention mechanism is similar to the fully connected layer in this scenario, which causes PVT to be unable to capture critical information. However, STFE can still extract important temporal and spatial features from a single packet, making STFE more efficient in this case. For longer sequences, the self-attention mechanism within the PVT module can use more context information to generate a stronger representation. Given the unique advantages of PVT and STFE, it is essential to use a dynamic weighting mechanism to effectively combine global and spatiotemporal features.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

V. Conclusion

This paper proposes a new encrypted traffic classification scheme called the ATVITSC. ATVITSC first presents a method to generate network flow images at the packet level, which preserves the original sequential nature of different packets after the image generation. Second, the ATVITSC splits the session image into multiple packet image sequences and extracts global and spatiotemporal features by using a multi-head self-attention mechanism and spatiotemporal feature extraction method. Next, ATVITSC applies a dynamic weighting mechanism to fuse these features to classify encrypted traffic effectively. Our scheme shows higher accuracy and generalization ability than existing methods in the tasks of encrypted malicious traffic classification, VPN encrypted traffic classification, Tor encrypted application classification, and mobile traffic classification.

However, ATVITSC also faces certain challenges. For example, it is still being determined whether ATVITSC can classify the encrypted traffic for dynamic datasets effectively. The encryption algorithms are updated constantly and the emergence of new algorithms may decrease the accuracy of ATVITSC in encrypted traffic classification. In the future, we will explore incremental learning methods to make the ATVITSC maintain good reusability in dynamic network environments. Meanwhile, we will introduce the latest multi-scale retention mechanism in the ATVITSC, which aims to speed up the time for classification and optimize its performance from the view of accuracy and efficiency.

References

[1] J. Xiang, N. Fulton, and S. Chong, "Relational analysis of sensor attacks on cyber-physical systems," in *Proc. IEEE 34th Comput. Secur. Found. Symp. (CSF)*, Jun. 2021, pp. 1–16.

[2] X. Lin, G. Xiong, and G. Gou, "ET-BERT: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proc. ACM Web Conf.*, New York, NY, USA, 2022, pp. 633–642, doi: 10.1145/3485447.3512217.

[3] R. Liu and X. Yu, "A survey on encrypted traffic identification," in *Proc. Int. Conf. Cyberspace Innov. Adv. Technol.*, New York, NY, USA, 2021, pp. 159–163, doi: 10.1145/3444370.3444564.

[4] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, "Identifying and discriminating between web and peer-to-peer traffic in the network core," in *Proc. 16th Int. Conf. World Wide Web*. New York, NY, USA, 2007, pp. 883–892, doi: 10.1145/1242572.1242692.

[5] R. T. Elmaghraby, N. M. A. Aziem, M. A. Sobh, and A. M. Bahaa-Eldin, "Encrypted network traffic classification based on machine learning," *Ain Shams Eng. J.*, vol. 15, no. 2, Feb. 2024, Art. no. 102361. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2090447923002502

[6] Z. Okonkwo, E. Foo, Q. Li, and Z. Hou, "A CNN based encrypted network traffic classifier," in *Proc. Australas. Comput. Sci. Week*. New York, NY, USA: ACM, Feb. 2022, pp. 74–83, doi: 10.1145/3511616.3513101.

[7] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 76–81, May 2019.

[8] J. Lan, X. Liu, B. Li, Y. Li, and T. Geng, "DarknetSec: A novel self-attentive deep learning method for darknet traffic classification and application identification," *Comput. Secur.*, vol. 116, May 2022, Art. no. 102663. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404822000621

[9] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for Internet of Things,"*IEEE Access*, vol. 5, pp. 18042–18050, 2017.

[10] P. Wang, X. Chen, F. Ye, and Z. Sun, "A survey of techniques for mobile service encrypted traffic classification using deep learning," *IEEE Access*, vol. 7, pp. 54024–54033, 2019.

[11] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 63–78, Jan. 2018.

[12] K. Al-Naami et al., "Adaptive encrypted traffic fingerprinting with bi-directional dependence," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, New York, NY, USA, Dec. 2016, pp. 177–188, doi: 10.1145/2991079.2991123.

[13] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: A flow sequence network for encrypted traffic classification," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1171–1179.

[14] T. Shapira and Y. Shavitt, "FlowPic: A generic representation for encrypted traffic classification and applications identification," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1218–1232, Jun. 2021.

[15] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Jul. 2017, pp. 43–48.

[16] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2017, pp. 712–717.

[17] M. Shen, J. Zhang, L. Zhu, K. Xu, and X. Du, "Accurate decentralized application identification via encrypted traffic analysis using graph neural networks," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2367–2380, 2021.

[18] T.-L. Huoh, Y. Luo, P. Li, and T. Zhang, "Flow-based encrypted network traffic classification with graph neural networks," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 2, pp. 1224–1237, Jul. 2023.

[19] Z. Wang, "The applications of deep learning on traffic identification," BlackHat USA, Las Vegas, LV, USA, Tech. Rep., 2015, pp. 1–10. [Online]. Available: https://www.blackhat.com/us-15/briefings.html#the-applications-of-deep-learning-on-traffic-identification

[20] M. Lotfollahi, R. S. H. Zade, M. J. Siavoshani, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Comput.*, vol. 24, no. 3, pp. 1999–2012, 2020, doi: 10.1007/s00500-019-04030-2.

[21] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, and S. Yu, "Identification of encrypted traffic through attention mechanism based long short term memory," *IEEE Trans. Big Data*, vol. 8, no. 1, pp. 241–252, Feb. 2022.

[22] Y. Kumano, S. Ata, N. Nakamura, Y. Nakahira, and I. Oka, "Towards real-time processing for application identification of encrypted traffic," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2014, pp. 136–140.

[23] K. Lin, X. Xu, and H. Gao, "TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of IIoT," *Comput. Netw.*, vol. 190, May 2021, Art. no. 107974. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621001067

[24] S. Zhu, X. Xu, H. Gao, and F. Xiao, "CMTSNN: A deep learning model for multiclassification of abnormal and encrypted traffic of Internet of Things," *IEEE Internet Things J.*, vol. 10, no. 13, pp. 11773–11791, Jul. 2023.

[25] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Netw.*, vol. 26, no. 1, pp. 35–40, Jan. 2012.

[26] S. Rezaei and X. Liu, "How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets," 2018, *arXiv:1812.09761*.

[27] A. Dosovitskiy et al., "An image is worth 16×16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[28] A. Vaswani et al., "Attention is all you need," in *Proc. NIPS*. Red Hook, NY, USA: Curran Associates, 2017, pp. 6000–6010.

[29] T. Völker, M. Tüxen, and E. P. Rathgeb, "The search of the path MTU with QUIC," in *Proc. Workshop Evol., Perform. Interoperability (QUIC)*, vol. 802. New York, NY, USA: Association for Computing Machinery, Dec. 2021, pp. 22–28, doi: 10.1145/3488660.3493805.

[30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

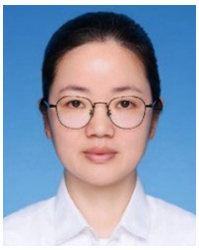
[31] S. Chaudhari, V. Mithal, G. Polatkan, and R. Ramanath, "An attentive survey of attention models," *ACM Trans. Intell. Syst. Technol.*, vol. 12, no. 5, pp. 1–32, Oct. 2021, doi: 10.1145/3465055.

[32] A. L. Maas. (2013). *Rectifier Nonlinearities Improve Neural Network Acoustic Models*. [Online]. Available: https://api.semanticscholar.org/CorpusID:16489696

[33] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[35] A. Graves, *Long Short-Term Memory*. Berlin, Germany: Springer, 2012, pp. 37–45, doi: 10.1007/978-3-642-24797-2_4.
[36] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.
[37] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2014.
[38] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[39] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[40] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Proc. Adv. Neural Inf. Process. Syst.*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Red Hook, NY, USA: Curran Associates, 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/f2925f97bc13ad2852a7a551802feea0-Paper.pdf
[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[42] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor traffic using time based features," in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, Feb. 2017, pp. 253–262.
[43] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proc. 2nd Int. Conf. Inf. Syst. Secur. Privacy*, 2016, pp. 407–414.
[44] T. Ede et al., "FlowPrint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, Jan. 2020, pp. 1–18.
[45] C. Catal, "Performance evaluation metrics for software fault prediction studies," *Acta Polytechnica Hungarica*, vol. 9, pp. 193–206, Jan. 2012.
[46] C. Liu, W. Wang, M. Wang, F. Lv, and M. Konan, "An efficient instance selection algorithm to reconstruct training set for support vector machine," *Knowl.-Based Syst.*, vol. 116, pp. 58–73, Jan. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705116304257



Ya Liu received the B.S. and M.S. degrees from the Mathematics Department, Anhui Normal University, in 2004 and 2007, respectively, and the Ph.D. degree in computer system architecture from Shanghai Jiao Tong University in 2013. She is currently an Associate Professor with the Department of Computer Science and Engineering, University of Shanghai for Science and Technology. Her research interests include cryptology, federated learning, the scalability and privacy protection of blockchain, and network security by network analysis. She serves as a member for IACR, CACR, and CCF.

Xiao Wang received the B.S. degree in engineering from the Huanghe University of Science and Technology in 2021 and the M.S. degree in engineering from the University of Shanghai for Science and Technology in 2024. His research interests include encrypted traffic classification.

Bo Qu received the B.S. and M.S. degrees in information security and computer science from Shanghai Jiao Tong University, China, in 2009 and 2012, respectively, and the Ph.D. degree in intelligent systems from Delft University of Technology, The Netherlands, in 2017. Before joining Guangdong University of Science and Technology, he worked with the Peng Cheng Laboratory as a Research Associate. He was also with Tencent Technology as a Researcher in applied research of network security. His research interests include network security by network analysis and network representation learning.

Fengyu Zhao received the B.S. and M.S. degrees in computer engineering from Nanjing University of Aeronautics and Astronautics in 1984 and 1989, respectively, and the Ph.D. degree in computer software and theory from Fudan University in 2010. From 1998 to 1999, he was a Visiting Scholar with Linköping University, Sweden. He is currently a Professor with the Department of Information and Intelligence Engineering, Shanghai Publishing and Printing College. His research interests include the design and analysis of software systems. He serves as a member for the Software Engineering Professional Committee of CCF.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on June 17,2025 at 02:47:44 UTC from IEEE Xplore. Restrictions apply.