Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.066367

ARTICLE

NetST: Network Encrypted Traffic Classification Based on Swin Transformer

Jianwei Zhang[1,*], Hongying Zhao[2], Yuan Feng[3,*], Zengyu Cai[2] and Liang Zhu[2]

[1]School of Software Engineering, Zhengzhou University of Light Industry, Zhengzhou, 450000, China
[2]School of Computer Science and Technology, Zhengzhou University of Light Industry, Zhengzhou, 450000, China
[3]School of Electronic Information, Zhengzhou University of Light Industry, Zhengzhou, 450000, China

*Corresponding Authors: Jianwei Zhang. Email: mailzjw@163.com; Yuan Feng. Email: fy@zzuli.edu.cn

Received: 07 April 2025; Accepted: 13 June 2025; Published: 30 July 2025

ABSTRACT: Network traffic classification is a crucial research area aimed at improving quality of service, simplifying network management, and enhancing network security. To address the growing complexity of cryptography, researchers have proposed various machine learning and deep learning approaches to tackle this challenge. However, existing mainstream methods face several general issues. On one hand, the widely used Transformer architecture exhibits high computational complexity, which negatively impacts its efficiency. On the other hand, traditional methods are often unreliable in traffic representation, frequently losing important byte information while retaining unnecessary biases. To address these problems, this paper introduces the Swin Transformer architecture into the domain of network traffic classification and proposes the NetST (Network Swin Transformer) model. This model improves the Swin Transformer to better accommodate the characteristics of network traffic, effectively addressing efficiency issues. Furthermore, this paper presents a traffic representation scheme designed to extract meaningful information from large volumes of traffic while minimizing bias. We integrate four datasets relevant to network traffic classification for our experiments, and the results demonstrate that NetST achieves a high accuracy rate while maintaining low memory usage.

KEYWORDS: Traffic classification; encrypted network traffic; Swin Transformer; network management; deep learning

1 Introduction

Network traffic classification plays a crucial role in network security and management. In the context of network services and supervision, tasks such as Internet resource allocation, network troubleshooting, and quality analysis are essential. However, the encryption of traffic data renders the original plaintext information inaccessible, posing a significant challenge to the normal operation and oversight of the network. Consequently, effectively analyzing and managing traffic without decrypting the data has become a critical issue that academics urgently need to address. Furthermore, by categorizing network traffic, operators can respond swiftly to diverse business needs, thereby enhancing service quality and user experience. Additionally, traffic classification is a vital component of intrusion detection systems, aiding in the identification of potential threats and ensuring system security.

With the increasing popularity of traffic encryption technologies, such as Tor and Virtual Private Networks (VPNs), which are designed to protect user privacy and anonymity, traditional traffic classification methods face new challenges. These technologies enable malware and cybercriminals to bypass conventional monitoring mechanisms, rendering traffic classification techniques that rely on port-based methods and deep packet inspection (DPI) less effective. On one hand, emerging technologies like port masquerading and

Network Address Translation (NAT) protocols diminish the efficacy of port-based classification methods; on the other hand, DPI depends on pattern and keyword recognition within packets [1], and encrypted traffic obstructs such recognition. Furthermore, the ongoing advancement of encryption technology complicates the ability of classification methods tailored for specific encrypted traffic to adapt to new environments or respond to evolving encryption policies. Consequently, identifying stable and concealed patterns within diverse encrypted traffic to achieve accurate and generalized traffic classification has become crucial for enhancing network security and improving network management efficiency. In recent years, to address these challenges, traffic classification techniques, particularly those utilizing deep learning, have made significant strides in the realm of encrypted traffic classification.

CNNs and Transformers have demonstrated exceptional performance in various areas of artificial intelligence, including computer vision [2–5] and natural language processing [6,7]. In recent years, researchers have introduced advanced techniques into the field of network traffic classification, achieving remarkable results. For instance, they have employed CNN-based methods to develop classification techniques based on visual representations [8,9], as well as Transformer-based, self-encoder, and multimodal classification approaches [10–13]. However, CNNs encounter limitations due to their small receptive fields, which hinder their ability to effectively process long sequences and capture distant dependencies. In contrast, Transformers exhibit significant advantages in traffic classification tasks, as they can handle long sequences and more effectively capture remote dependency information more effectively. However, the Transformer architecture has quadratic complexity [14], and when processing large-scale data, the demand for computational resources increases significantly, leading to a decrease in efficiency. Although Transformer can effectively process long sequences of dependencies, when the length of the network traffic data is very long, Transformer may still face processing efficiency and memory usage problems. Therefore, this paper introduces the Swin Transformer architecture into the field of network traffic classification. Swin Transformer utilizes hierarchies and windows instead of long sequences, a feature that enables it to capture both local details and global context information with low computational complexity and superior performance accuracy. The primary contributions of this paper are as follows:

1. Construct a multilevel flow matrix representation of raw traffic. This representation generates a two-dimensional matrix from raw traffic data, capable of illustrating information across byte, packet, and flow levels. It offers a comprehensive view of traffic features, in contrast to most existing methods that only intercept a fixed number of bytes at the beginning of the flow.
2. Introducing the four-stage Swin Transformer architecture in the vision domain for network traffic classification, we propose the three-stage Net Swin Transformer (NetST). By preserving the benefits of window attention and the shift mechanism, the original four-stage hierarchical structure is streamlined into three stages. This modification reduces the downsampling depth of late feature maps, enhances the retention of mesoscale contextual information, and improves the ability to detect key traffic behavioral patterns. NetST significantly decreases the number of model parameters and accelerates inference speed while maintaining classification accuracy, resulting in higher deployment efficiency.
3. In the field of image classification, several advanced models have been extensively researched and applied. This paper introduces these advanced models from image classification into the domain of network traffic classification, demonstrating their potential in this new context. Furthermore, the paper provides a comprehensive comparison of various advanced image classification algorithms through experimental analysis, offering insights into cross-domain learning for these models.

2 Related Work

In order to enhance the efficiency and accuracy of network traffic classification, this paper proposes a more comprehensive scheme for traffic feature representation, drawing on the Swin Transformer architecture utilized in image classification. This section will focus on related research in four areas: commonly used datasets and traffic representation schemes in cryptographic network traffic classification, prevalent classification methods, and the application of image classification algorithms in network traffic classification.

2.1 Datasets and Network Traffic Representation Schemes

In recent years, research in the field of network encrypted traffic classification has encountered several challenges. First, current studies often rely on one or two classes of datasets, resulting in insufficient coverage of network scenarios and limited generalization capabilities of the models. These issues can adversely affect the performance and adaptability of the models in various environments. Second, the traffic feature representation schemes are not yet comprehensive enough. Due to limitations in feature extraction, models may struggle to effectively capture the key characteristics that differentiate various traffic classes, leading to decreased classification accuracy. Additionally, this reliance on specific datasets can hinder the model's ability to transfer across different datasets. Statistical features (e.g., number of packets, packet size, minimum interval time, etc.) often fail to capture the specific content of the packets. This lack of contextual information regarding the actual content can lead to inaccurate classification results when these statistical features are employed. In contrast, combining header and payload data for traffic classification allows for the extraction of more comprehensive traffic features, thereby enhancing classification accuracy. However, many studies have not anonymized IP addresses and MAC addresses. When these addresses are used directly for classification, the model tends to rely excessively on these features, which negatively impacts the accuracy of the classification results.

To address the limitations of existing studies, which often rely on small datasets and lack comprehensive traffic characterization schemes, this study integrates datasets from five distinct network traffic domains for a more in-depth analysis. We not only extract the headers and payloads from the raw traffic but also anonymize the IP and MAC addresses. The header and payload contain critical information for data transmission; the header provides metadata about the network communication (e.g., source address, destination address, port number), while the payload contains the actual transmitted data. By combining these two components, we can achieve a more comprehensive traffic characterization, thereby enhancing the accuracy of classification. In this paper, we analyze the number of datasets and feature representation schemes employed by common methods in the field of traffic classification, as well as the datasets and input features utilized in this study. The relevant statistical information is presented in Table 1.

Table 1: Dataset and input features used by different methods

Dataset	Method	Input features
ISCXVPN2016	[15]	Relative time and data packet length
	[16]	Delete the IP and mac address, 32 × 32 image
	[17]	Statistical features on the data package
ISCXVPN2016, ISCXTor2016	[18]	Packet size, arrival time
	[19]	Remove IP and mac addresses, top 15 packets
	[20]	30 × 30 image, and the IP were not anonymous

(Continued)

Table 1 (continued)

Dataset	Method	Input features
ISCXVPN2016, USTC-TFC2016	[21]	No head, 28 × 28 image, and Payload 784 B
	[22]	Packets 10, Payload 784 B, and IP were not anonymous
CrossPlatform-Android, CrossPlatform-iOS	[23]	Packets 6, Payload 60 B, and IP were not anonymous
	[24]	Packets N, Payload 256 B, and IP were not anonymous
CrossPlatform-Android, CrossPlatform-iOS, USTC-TFC2016, ISCXVPN2016, ISCXTor2016	NetST (ours)	40 × 40 image, Payload B, and IP were anonymous

2.2 *Encryption Traffic Classification Method*

2.2.1 Network Traffic Classification Based on Traditional Methods

Traditional traffic classification methods fall into two main categories: port-based and DPI techniques. The port-based network traffic classification technique represents one of the earliest classification approaches. Since each application is assigned a unique port number by the Internet Assigned Numbers Authority (IANA) (e.g., FTP protocol applications default to port 21, SSH protocol applications to port 22), this technique distinguishes network traffic types by analyzing transport layer port information. Cheng and Wang [25] proposed a port-number-connection-pattern method for differentiating server application traffic. However, emerging technologies like port camouflage techniques and NAT protocols have rendered dynamic port traffic unmappable to specific applications, significantly reducing classification accuracy.

Deep Packet Inspection (DPI)-based approaches classify traffic by analyzing the application-layer payload of packets. DPI identifies various protocols using predefined patterns and regular expressions [1]. However, DPI is ineffective against emerging protocols, as it requires periodic updates to its pattern library to accommodate new protocols. Furthermore, DPI struggles to efficiently handle encrypted traffic because the content of encrypted data cannot be analyzed directly.

2.2.2 Network Traffic Classification Based on Machine Learning Methods

To address the limitations of traditional network traffic classification techniques in handling encrypted traffic, researchers have integrated machine learning approaches by labeling traffic data with specific characteristics and training models with these annotated datasets, thereby enabling more accurate pattern matching and identification. Machine learning-based classification algorithms primarily fall into two categories: supervised and unsupervised learning. In the realm of supervised learning, Kong et al. [26] developed an Attack Traffic Identification System (ATIS) applying Support Vector Machines (SVM) for IP network traffic identification and multi-attack traffic detection, whereas He [27] introduced a weighted feature SVM method that mitigates sample distribution bias, enhances computational efficiency, and improves classification accuracy through better generalization. Within unsupervised learning paradigms, Chen et al. [28] proposed an enhanced density peak clustering algorithm for cryptographic malware traffic detection, while Celik et al. [29] conducted comparative evaluations of K-means, One-Class SVM (OCSVM), Least Squares Anomaly Detection (LSAD), and K-nearest neighbors (KNN) algorithms.

2.2.3 Network Traffic Classification Based on Deep Learning Method

Although machine learning algorithms address the shortcomings of port-based and deep packet detection techniques to a certain extent, their performance is limited by the construction of the feature set, which directly affects the accuracy of recognition. In contrast, deep learning methods can directly input data into neural networks for training after data preprocessing, and the advantage of this method is that it can automatically extract features, reducing the workload of manual feature extraction. Therefore, deep learning not only improves the efficiency of network traffic classification, but also provides a new direction for research in this field.

The most classical deep learning-based traffic classification model is to use CNN to extract the spatial features of traffic, the most typical one is proposed by Wang et al. [30] to convert the traffic data into grayscale images, and use CNN to classify the malicious traffic without manually designing the feature set; after that, Wang et al. [31] proposed an end-to-end based on a one-dimensional convolutional neural network end-to-end encrypted traffic classification method, which integrates the three key steps of traffic feature extraction, feature selection and classification into a unified end-to-end framework and automatically learns the nonlinear relationship between the original input data and the expected output, which is the first time that an end-to-end method has been applied to the encrypted traffic classification task. Okonkwo et al. [8] designed an 11-layer CNN and trained it using a series of images generated from metadata of encrypted traffic; and literature [32] treats each packet as a two-dimensional picture and connects multiple packets (pictures) are connected to form video frames that constitute a 3D video using three convolutional layers and three pooling layers to extract high-level features. Since CNN is weak to the interaction information of different time steps when capturing network flow features, its effect is limited when used alone, to solve this problem, researchers combined with the RNN model, by first extracting the spatial features of the flow with CNN, and then capture the temporal features of the packets by LSTM or RNN [33–37], so as to effectively obtain the flow feature vector, this combination method can to some extent improve the accuracy of feature representation.

2.2.4 Application of Image Classification Algorithm in the Field of Network Traffic Classification

The application of image classification algorithms in the field of network traffic classification is mainly based on the image classification method in computer vision, by converting the network traffic features into the form of "image" to carry out classification, this method combines deep learning and network traffic analysis techniques, and utilizes deep neural networks (such as Convolutional Neural Networks, CNN) and other. This approach combines deep learning and network traffic analysis techniques, and utilizes deep neural networks (e.g., convolutional neural networks, CNN) and other image processing methods to efficiently classify network traffic data. In recent years, most researchers have converted the data of network traffic into image format through certain conversion methods, and then applied image classification algorithms, which are able to deal with multidimensional and large-scale network traffic data, especially when the traffic data volume is large and the changes are complex, and the model is able to show strong robustness. The most representative of them are based on convolutional neural network, based on Transform architecture, although CNN is good at capturing the local features of the image [31,32], it does not have the advantage of capturing the global contextual information, while Transformer's self-attention mechanism is able to capture the dependencies between different positions in the input sequence, which can help the model to understand the relationship between different parts of the image. relationships between them Therefore, researchers have proposed to introduce the Transformer architecture into the field of traffic classification [34–36] and have achieved some research results. However, due to the model efficiency problem of the Transformer architecture due to secondary complexity, it is difficult to meet the real-time requirements in the field of network traffic analysis, for this reason, this paper proposes to apply the Swin Transformer architecture to the task of network traffic categorization. Swin Transformer [14] is a highly efficient variant of Transformer for visual tasks. An efficient variant of Transformer, it significantly reduces the computational effort by introducing a local windowing mechanism, which enhances the application of Transformer in the visual domain.

Unlike Vision Transformer (ViT), which is based on global image block processing, Swin Transformer adopts the sliding window approach to image segmentation, a feature that enables it to capture both local details and global contextual information, and has become one of the mainstream technologies in the field of computer vision by virtue of its low computational complexity and excellent performance accuracy. With low computational complexity and excellent performance accuracy, Swin Transformer has become one of the mainstream technologies in the field of computer vision. In this paper, we use the Swin Transformer architecture to study its applicability in the field of network traffic classification.

3 NetST Traffic Classification Model

NetST method consists of three main steps. First, in the Input Layer, the captured raw traffic (PCAP packets) is transformed into a multilevel flow matrix (MFM), which is represented as a two-dimensional grayscale image. Next, the traffic data processed in the Input Layer is input into the NetST layer for model training. Finally, the data trained in the NetST layer is passed through the Output Layer, where the Softmax function is applied to generate the final classification results. The model architecture is illustrated in Fig. 1.

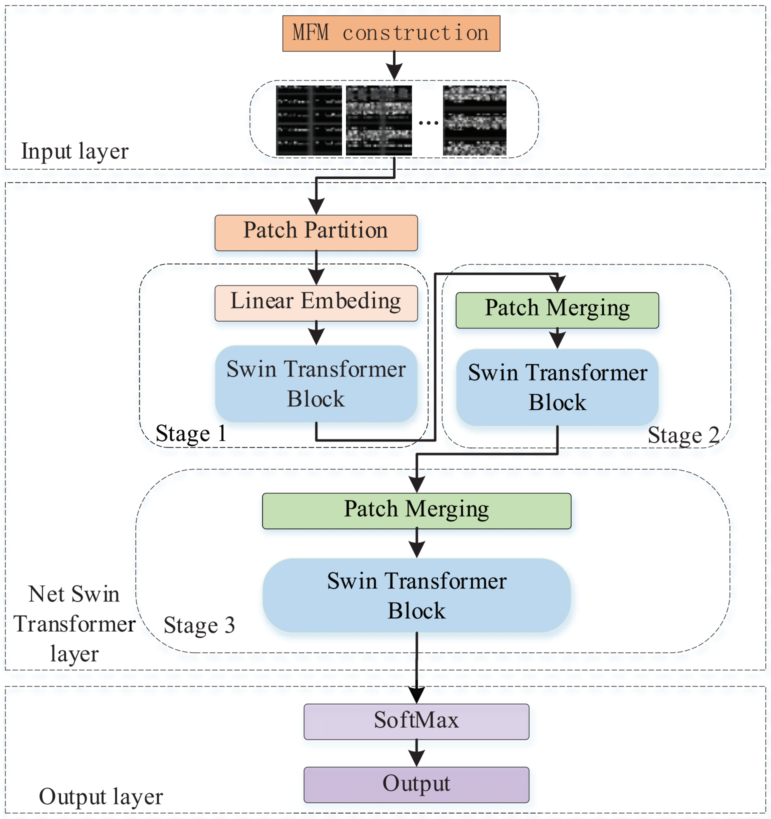


Figure 1: Architecture of network encrypted traffic classification method based on Swin Transformer

3.1 Input Layer

The NetST-based network traffic classification method models raw traffic by creating a MFM generated from raw packet bytes and formatted to contain traffic information at different levels of granularity through a formatting matrix. First, the raw traffic is split into streams based on IP address, port number, and protocol type; then, to avoid introducing bias interference, the Ethernet header of the stream is removed, the port number is set to zero, and the IP is replaced with a random address but its direction is retained; finally, M neighboring packets in the stream are captured and formatted into a two-dimensional matrix of size $H \times W$ as a representation of the stream. This is shown in Fig. 2. The traffic representation matrix includes byte-level, packet-level and data-flow level information, as detailed below:

1. Byte-level: Each matrix row exclusively encodes a specific traffic byte type, with explicit partitioning into header rows (protocol metadata) and payload rows (application data).
2. Packet-level: Individual packets are represented through dual matrices—a header matrix and payload matrix—constituting packet-level matrices with dimensions $(H/M) \times W$.
3. Stream-level: Ordered packet sequences form streams through spatial stacking of M consecutive packet-level matrices along the vertical axis, generating final Multi-level Flow Representation matrices.

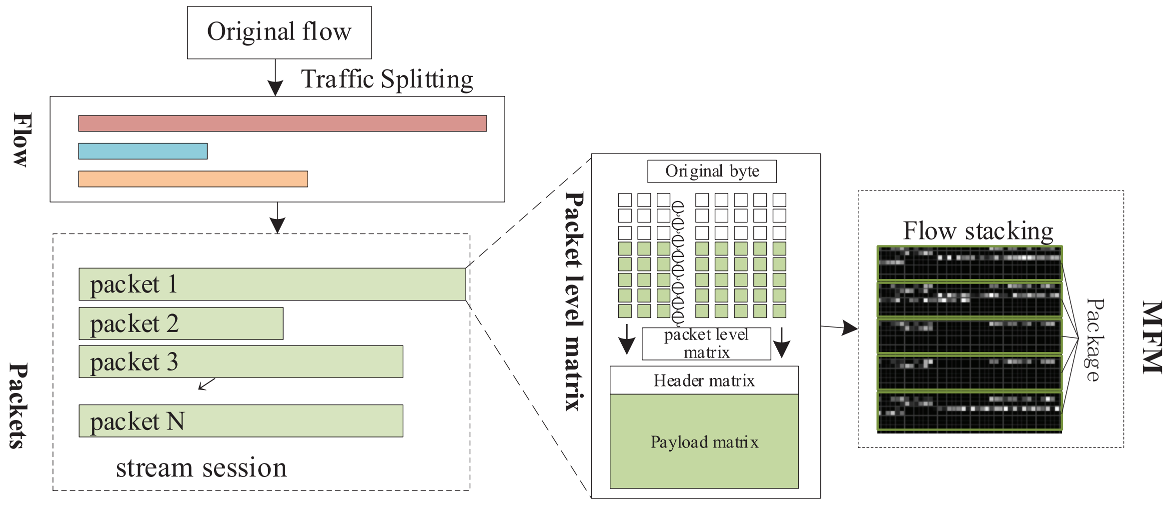


Figure 2: Multi-level flow matrix construction

We follow previous research [10], the MFM is designed as a fixed-size 40 × 40 matrix for normalizing the protocol headers and payloads of the first 5 packets in a characterized network flow. Each packet is encoded into 8 lines:

1. The header section consists of 2 rows, totaling 40 bytes: the IP layer (20 bytes) and the TCP/UDP layer (20 or 8 bytes), along with optional headers. These components are specifically reserved to ensure that essential network and transport layer features, such as the IP Time to Live (TTL) and TCP flag bits, are encoded independently, thereby preventing confusion with load noise.
2. Payload section (6 lines × 40 bytes): truncated or zero-complemented to 240 bytes, emphasizing initial application layer features such as the HTTP request header and TLS handshake parameters. Experiments have demonstrated that the first 240 bytes are highly discriminative for classification.
3. Dimension design: 5 packets are stacked vertically ($5 \times 8 = 40$ rows), with each row fixed at 40 bytes, which is compatible with the maximum standard header length of IP and TCP. This configuration forms a

40 × 40 matrix. The design accommodates models such as CNN by maintaining a fixed input dimension while balancing information integrity and computational efficiency through protocol-layered coding and the extraction of load-critical segments.

3.2 Net Swin Transformer Layer

The input image is initially divided into several small patches by the Patch Partition layer. Each patch is then mapped to a high-dimensional feature space by the Linear Embedding layer, forming the initial token sequence. In Stage 1, multiple Swin Transformer Blocks are stacked for feature extraction. In Stages 2 and 3, the Patch Merging layer is employed at the beginning of each stage for downsampling neighboring patches and channel splicing, thereby constructing multi-scale hierarchical features. Within each Swin Transformer Block, the Window-based Multiple Self-Attention (W-MSA) and Shift-Window Self-Attention (SW-MSA) mechanisms are alternately utilized. This approach significantly reduces computational complexity while enhancing information interaction between windows. Finally, the multi-scale features are integrated and utilized for subsequent downstream classification tasks.

Swin Transformer Block

Swin Transformer consists of two consecutive Swin Transformer Blocks, which are used to improve the computational efficiency and ultimately the classification accuracy through Multi-head Self-Attention of Windows and Shifted Windows. The architecture of the Swin Transformer Block is illustrated in Fig. 3.

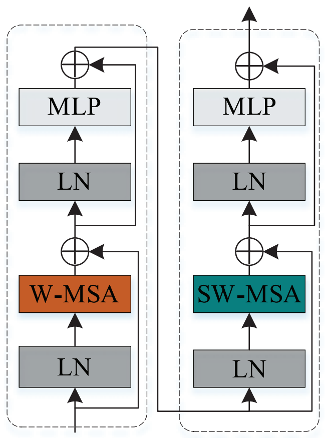


Figure 3: Swin Transformer block

(1) Windows Multi-Head Self-Attention (W-MSA)

In the visual Transformer model, the traditional Multi-Head Self-Attention (MSA) mechanism computes the self-attention weights for each pixel in the feature map in relation to all other pixels. This results in a computational complexity of $O(h^2w^2.C)$, where h, w, and C represent the height, width, and depth of the feature map, respectively, as illustrated in Fig. 4a. Therefore, the Swin Transformer introduces the Windows Multi-Head Self-Attention (W-MSA) mechanism to optimize computation. W-MSA segments the feature map into multiple non-overlapping sub-regions (windows) based on a window size of M × M. Within each window, self-attention is computed independently, as illustrated in Fig. 4b. This method reduces the computational complexity from global $O(h^2w^2.C)$ to $O(\frac{h}{M}.\frac{W}{M}.(M^2)^2.C) = O(M^2hw.C)$ within each window, thereby achieving linear computational complexity and significantly decreasing the overall computational load.

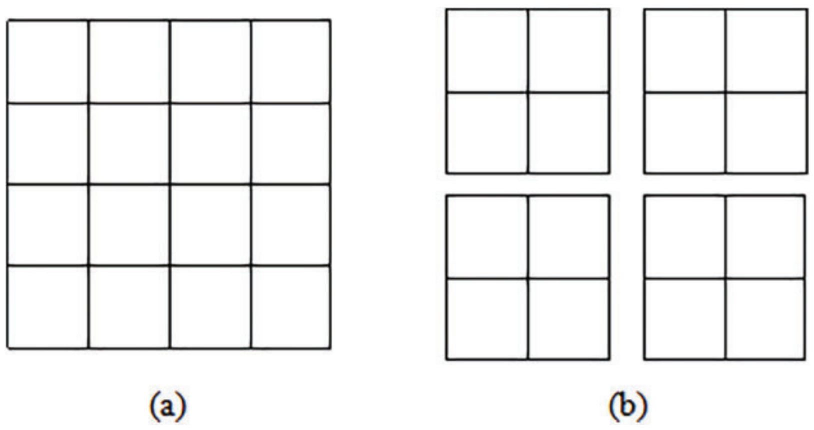


Figure 4: Multi-Head Self-Attention (a) and Windows Multi-Head Self-Attention (b)

(2) Shifted Windows Multi-Head Self-Attention (SW-MSA)

With the W-MSA module, each window computes self-attention solely within its own boundaries, resulting in no direct information transfer between different windows. To address this limitation, the SW-MSA module is introduced, which offsets windows in neighboring layers. As illustrated in Fig. 5, when the standard W-MSA is applied in layer L, layer $L + 1$ shifts the window to the right and downward by $\frac{M}{2}$ pixels each. This adjustment allows patches that were originally separated into different windows to overlap in the new window, thereby facilitating cross-window information exchange. For instance, a window in the first row and second column after the offset (e.g., 2×4) enables information sharing between two windows in the first row of layer L, while a window in the second row and second column (e.g., 4×4) allows for information sharing among four windows. By utilizing the W-MSA and SW-MSA modules interchangeably, the efficiency of local self-attention computation is preserved while overcoming the challenge of limited information transfer between windows.

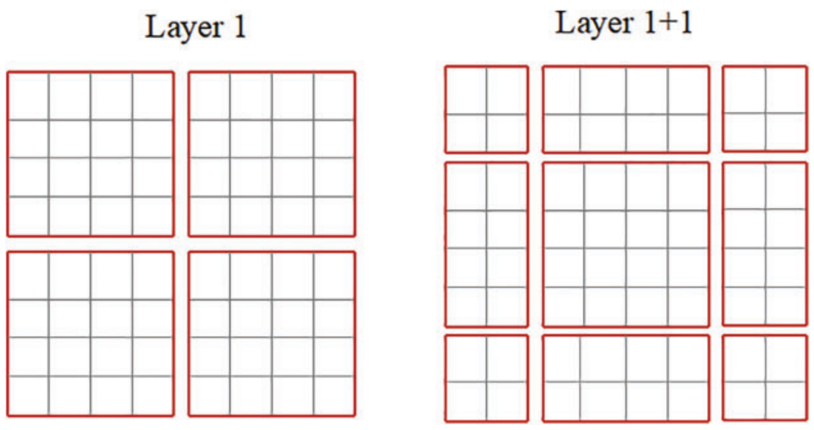


Figure 5: The SW-MSA mechanism

Through patch segmentation, linear embedding, and hierarchical construction of images, the Swin Transformer achieves CNN-like multi-scale feature extraction. Additionally, computational complexity is significantly reduced by employing W-MSA, while the introduction of SW-MSA effectively facilitates information exchange across windows. These innovations render the Swin Transformer highly effective in representation and transfer learning, all while maintaining high efficiency.

3.3 Patch Merging

In the Swin Transformer, each stage, except for the first, begins with downsampling through the Patch Merging layer. As illustrated in Fig. 6, this layer divides the input feature map into 2 × 2 non-overlapping regions, extracts the corresponding pixels from each region, and concatenates them along the depth dimension to create a new feature map with four times the number of channels. Subsequently, the number of channels is linearly mapped from 4C to 2C using LayerNorm normalization and a fully connected layer. This process not only halves the spatial size of the feature map but also doubles the number of channels, effectively constructing a hierarchical representation while reducing computational complexity.

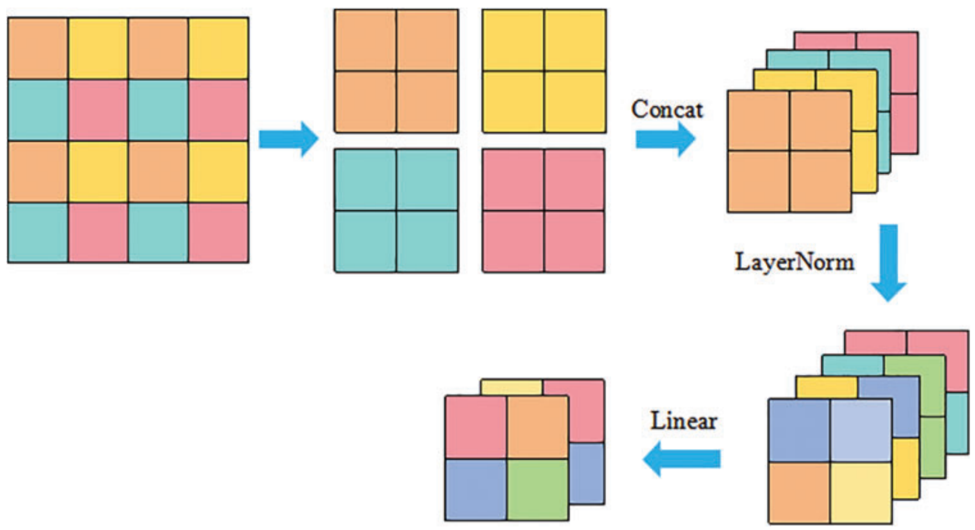


Figure 6: Patch merging layer feature map processing

In this paper, the input image is denoted as $X \in \mathbb{R}^{40\times 40}$. First, the image is divided into several small, non-overlapping regions (patches), each with a spatial size of 2 × 2 pixels. Consequently, 20 patches can be created along both the vertical and horizontal directions, resulting in a total of 400 patches generated. For each patch P_i (where $i = 1, 2, \ldots, 400$) it is represented as a one-dimensional vector $\mathrm{x_i} \in \mathbb{R}^4(\mathrm{x_i} = \mathrm{vec}(P_i))$, where $\mathrm{vec}(.)$ denotes the arrangement of the two-dimensional patches in a fixed order). This is followed by the introduction of a linear embedding layer characterized by a parameter matrix $\mathrm{W_e} \in \mathbb{R}^{\mathrm{d}\times 4}$ (where d is the embedding dimension), which maps the flattened vectors to a high-dimensional feature space $\mathrm{Z_i} = \mathrm{W_e x_i} \in \mathbb{R}^{\mathrm{d}}$. In this manner, the entire image is transformed into a sequence of tokens, each with a length of 400, following patch segmentation and embedding. The dimension of each token is denoted as d. The Patch Merging layer reduces computational effort by halving the spatial size while simultaneously enhancing feature representation through the reorganization of channel information and linear transformation. This process provides hierarchical input features for the subsequent Swin Transformer Block module.

The hierarchical processing flow of our proposed model achieves progressive equilibrium between spatial information compression and semantic enhancement through phased feature transformations. The input layer ingests single-channel 40 × 40 grayscale images, which undergo patch embedding to partition the input into 2 × 2 non-overlapping patches (resolution reduced to 20 × 20), while simultaneously expanding channel dimensions to 192 via linear projection, thereby establishing an initial high-dimensional feature representation (400 × 192). Stage 1 maintains the 20 × 20 resolution through local window attention mechanisms, conducting fine-grained feature modeling within 192-channel space. Stage 2 implements patch merging to halve spatial resolution to 10 × 10 while doubling channels to 384, effectively expanding receptive fields for mid-level pattern capture. Stage 3 further compresses resolution to 5 × 5 with 768 channels, employing global attention to aggregate high-level semantic representations. The output layer ultimately maps these 5 × 5 × 768 features to n-dimensional class probabilities via global pooling. This architectural progression systematically decouples spatial resolution (40 → 5) from channel dimensions (1 → 768), achieving efficient semantic abstraction with local detail preservation, while providing multi-scale feature support for small-image classification. Critical tensor transformations are quantitatively summarized in Table 2.

Table 2: Key tensor changes during the training phase

Processing stage	Tensor shape	Spatial resolution	Channels
Input layer	(B, 1, 40, 40)	40 × 40	1
Patch Embedding	(B, 400, 192)	20 × 20	192
Stage 1	(B, 400, 192)	20 × 20	192
Stage 2	(B, 100, 384)	10 × 10	384
Stage 3	(B, 25, 768)	5 × 5	768
Output layer	(B, n)	–	n

4 Experiment

In this section, we conduct four tasks related to encrypted traffic classification (Section 4.1) to demonstrate the effectiveness of NetST in addressing various encryption scenarios. Subsequently, we compare our model with nine different methods (Section 4.2) and analyze the model's hyperparameters (Section 4.3). Finally, we further examine the impressive performance achieved by NetST (Section 4.4).

4.1 Experimental Setup

All experiments were conducted in a consistent environment using the following experimental equipment configuration: AMD® Epyc 7302 16-core processor (×64), NVIDIA GeForce RTX 3090, and Ubuntu 22.04.5 LTS.

4.1.1 Dataset

In order to evaluate the effectiveness and generalization ability of the NetST model, this paper collects raw data samples from five publicly available datasets for experimentation. It conducts four encrypted traffic classification tasks across these datasets and divides each dataset into a training set, a validation set, and a test set in a ratio of 8:1:1, as shown in Table 3.

Table 3: Dataset information

Assignment	Dataset	Class	Training set	Validation set	Test set
Task 1	CrossPlatform-Android	181	44,700	4655	4656
	CrossPlatform-iOS	124	40,379	4203	4205
Task 2	USTC-TFC2016	20	37,323	6692	6692
Task 3	ISCXVPN2016	7	13,281	1383	1384
Task 4	ISCXTor2016	8	11,655	1456	1458

Task 1: Categorization of Cryptocurrency Applications.

CrossPlatform-Android category comprises 196 applications, while the CrossPlatform-iOS category includes 215 applications. The applications for both iOS and Android were sourced from the top 100 applications in the United States, China, and India.

Task 2: Classification of Cryptocurrency-Related Malware.

The USTC-TFC2016 dataset comprises a collection of encrypted network traffic generated by both malware and benign applications. It encompasses ten distinct categories of benign traffic and ten categories of malicious traffic.

Task 3: Classification of Applications Utilizing Virtual Private Network (VPN) Encryption.

ISCXVPN2016 dataset comprises encrypted communication traffic transmitted via VPN tunnels. VPNs are frequently employed to bypass censorship and conceal geographical locations through protocol obfuscation.

Task 4: Classification of Encryption Applications Utilizing Onion Routing (Tor).

ISCXTor2016 dataset encompasses application traffic that employs encrypted communication via the Tor. This method introduces an additional layer of obfuscation to the communication by utilizing a distributed routing network.

4.1.2 Assessment Indicators and Experimental Parameterization

Evaluation Metrics: To comprehensively assess the performance of various models, this paper utilizes Accuracy (AC), Precision (PR), Recall (RC), and the F1 score (F1) as evaluation criteria. The formulas for these calculations are as follows:

$$\text{Accurary} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

TP represents the number of normal data points accurately classified by the model, while *TN* denotes the number of abnormal data points correctly identified. *FP* refers to the number of anomalies incorrectly classified as normal data, and *FN* indicates the number of normal data points misclassified as anomalous.

Experimental Parameters: The number of training rounds is set to 100, with a batch size of 64. The AdamW optimizer is employed, and the learning rate is dynamically adjusted using the ReduceLROnPlateau scheduler, with a base learning rate of 0.0001. The proposed method is implemented using the PyTorch deep learning framework on a NVIDIA GeForce RTX 3090 GPU server, utilizing a single graphics card for training. The parameter settings are presented in Table 4.

Table 4: Experimental parameter setting

Parameter	Set up
Batch size	64
Epoch	100
Scheduler	ReduceLROnPlateau
Optimizer	Adam
Criterion	CrossEntropyLoss
Patch_size	2
Mlp_ratio	1
Hidden_dim	192
Layers	(2,6,2)
Heads	(3,6,12)

4.2 Comparison Experiment

4.2.1 Comparison of Existing Encrypted Traffic Classification Algorithms

In this paper, we present the experimental results from reference [10], which are analyzed in comparison with several commonly used methods in the field of network traffic classification. These methods include FS-net [1], YaTC [10], ET-BERT [12], 2D-CNN [31], Deeppacket [38] and PERT [39]. The experimental results, obtained from representative datasets such as USTC-TFC2016, ISCXVPN2016, ISCXTor2016, and CICIoT2022, demonstrate that the NetST model proposed in this study achieves significant improvements in both AC and F1 score F1. NetST achieves an accuracy of 99.91% and an F1 score of 99.03% on the USTC-TFC2016 dataset, representing a 2.05% improvement in accuracy compared to YaTC, which has the next best performance (Accuracy: 97.86%, F1: 97.86%). On the ISCXVPN2016 dataset, NetST attains an accuracy of 98.48% and an F1 score of 98.15%, outperforming YaTC (Accuracy: 98.07%, F1: 87.74%) with a 0.41% improvement in accuracy and a 10.41% improvement in F1 score. In the ISCXTor2016 dataset, NetST demonstrates exceptional performance with an accuracy of 99.93% and an F1 score of 99.92%, significantly surpassing other methods. Additionally, on the CICIoT2022 dataset, NetST performs admirably with an accuracy of 99.85% and an F1 score of 99.78%, which is considerably higher than the suboptimal model YaTC (Accuracy: 96.58%, F1: 96.58%). The experimental results clearly indicate that the NetST model outperforms existing mainstream methods in terms of both accuracy and F1 score across various network traffic classification datasets, as illustrated in Table 5.

Table 5: Performance comparison of existing models on different datasets

	USTC-TFC2016		ISCXVPN2016		ISCXTor2016		CICIoT2022	
	AC	**F1**	**AC**	**F1**	**AC**	**F1**	**AC**	**F1**
Deeppacket	0.8849	0.8883	0.8021	0.8017	0.3681	0.2681	0.8828	0.8808
2D-CNN	0.9226	0.9205	0.8126	0.8064	0.3462	0.3366	0.9007	0.9000
FS-net	0.8705	0.9602	0.8764	0.873	0.5203	0.5164	0.8537	0.853
PERT	0.9663	0.9664	0.8862	0.8861	0.8022	0.7999	0.9052	0.9049
ET-BERT	0.9695	0.9695	0.8774	0.8747	0.6538	0.6498	0.9035	0.9031
YaTC	0.9786	0.9786	0.9807	0.8774	0.9972	0.9972	0.9658	0.9658
NetST	0.9991	0.9903	0.9848	0.9815	0.9993	0.9992	0.9985	0.9978

The NetST model demonstrates exceptional performance across all test datasets, as illustrated in Fig. 7. On the USTC-TFC2016 dataset, NetST achieves the highest accuracy and F1 score, showcasing its superior classification capabilities. Similarly, on the CICloT2022 dataset, NetST performs remarkably well, further validating its strong generalization ability. Additionally, NetST excels in accuracy and F1 scores on both the ISCXVPN2016 and ISCXTor2016 datasets. In contrast, other models such as YaTC, ET-BERT, PERT, FS-net, 2D-CNN, and Deeppacket exhibit relatively weaker performance on certain datasets. Overall, the NetST model surpasses its competitors in terms of accuracy and F1 scores, particularly across a diverse range of datasets. These results indicate that NetST possesses significant advantages in addressing the task of network encrypted traffic classification, providing a more accurate and efficient classification capability across various datasets and application scenarios.

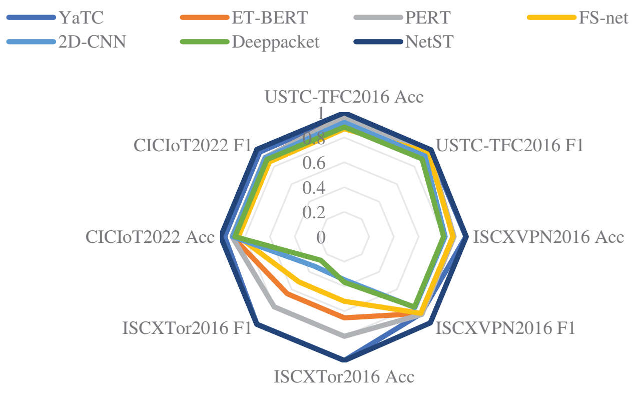


Figure 7: Star chart comparison of different model performance

4.2.2 Comparison with Advanced Classification Algorithms

To verify the effectiveness of the model proposed in this paper for the task of encrypted traffic classification, we designed and conducted several sets of comparative experiments. In these experiments, we compared the models presented in this paper with algorithms that have demonstrated strong performance in image classification tasks. These algorithms are of significant reference value due to their broad applicability and exceptional performance. To ensure the fairness and accuracy of the experimental results, all models were tested using the same dataset, preprocessing steps, and evaluation metrics. The results indicate that the model proposed in this paper outperforms the other models across all evaluation metrics in the encrypted traffic classification task, thereby confirming its superiority in this domain. The experimental results are presented in Tables 6 and 7.

Table 6: Results of datasets CrossPlatform-Android, CrossPlatform-iOS and USTC-TFC2016

	CrossPlatform-Android				**CrossPlatform-iOS**				**USTC-TFC2016**			
	AC	**PR**	**RC**	**F1**	**AC**	**PR**	**RC**	**F1**	**AC**	**PR**	**RC**	**F1**
ResNet101	0.9682	0.9676	0.9670	0.9656	0.9767	0.9733	0.9746	0.9718	0.9991	0.9903	0.9906	0.9903
EfficientNet	0.9680	0.9684	0.9660	0.9658	0.9757	0.9771	0.9723	0.9698	0.9981	0.9864	0.9882	0.9869
ViT	0.9691	0.9691	0.9679	0.9666	0.9743	0.9697	0.9724	0.9700	0.9990	0.9935	0.9920	0.9927
NetST	0.9774	0.9758	0.9749	0.9749	0.9769	0.9701	0.9734	0.9710	0.9995	0.9964	0.9959	0.9960

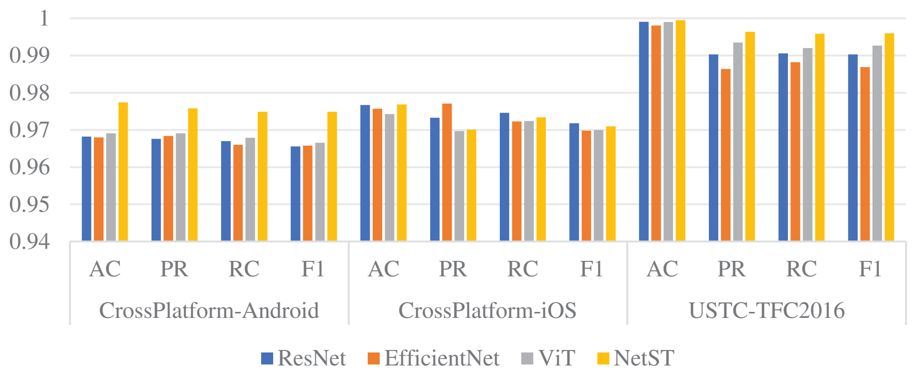
Table 7: Results of datasets ISCXVPN2016, ISCXTor2016

	ISCXVPN2016				**ISCXTor2016**			
	AC	**PR**	**RC**	**F1**	**AC**	**PR**	**RC**	**F1**
ResNet101	0.9812	0.9738	0.9708	0.9722	0.9904	0.9904	0.9895	0.9898
EfficientNet	0.9781	0.9621	0.9585	0.9602	0.9986	0.9988	0.9982	0.9985
ViT	0.9776	0.9764	0.9648	0.9704	0.9986	0.9986	0.9986	0.9986
NetST	0.9863	0.9780	0.9815	0.9797	0.9993	0.9994	0.9990	0.9992

The NetST model demonstrates strong performance across several datasets. On the CrossPlatform-Android, CrossPlatform-iOS, and USTC-TFC2016 datasets, NetST's Accuracy (AC), Precision (PR), Recall (RC), and F1 scores are either higher than or comparable to those of other models. Notably, on the USTC-TFC2016 dataset, the Accuracy reached 0.9995, and the F1 score was 0.9960. Additionally, the NetST model performs well on the ISCXVPN2016 and ISCXTor2016 datasets, particularly on the ISCXTor2016 dataset, where all metrics are nearly perfect, with an accuracy rate of 0.9993 and an F1 score of 0.9992. The USTC-TFC2016 and ISCXTor2016 datasets exhibit higher training accuracies, primarily because they contain more consistent and easily distinguishable features, enabling the model to effectively extract relevant characteristics and enhance performance. In contrast, the CrossPlatform dataset encompasses multiple platforms with diverse and complex features, which complicates the classification task and adversely affects the model's training outcomes. Consequently, a more specialized and distinct dataset leads to improved model performance in terms of accuracy and other metrics. As illustrated in Fig. 8.

4.3 Comparison of Various Optimizers and Schedulers

Table 8 presents a comparison of ACC among different optimizers (Adam and AdamW) and learning rate scheduling strategies (ReduceLROnPlateau, MultiStepLR, and CosineAnnealingLR) across five datasets: CrossPlatform-Android, CrossPlatform-iOS, USTC-TFC2016, ISCXVPN2016, and ISCXTor2016. The results indicate that the combination of Adam with ReduceLROnPlateau yields strong performance. The Adam optimizer effectively accelerates convergence through its adaptive learning rate, while the ReduceLROnPlateau scheduler automatically reduces the learning rate when the model's training stagnates. This mechanism helps prevent overfitting and further enhances model performance.



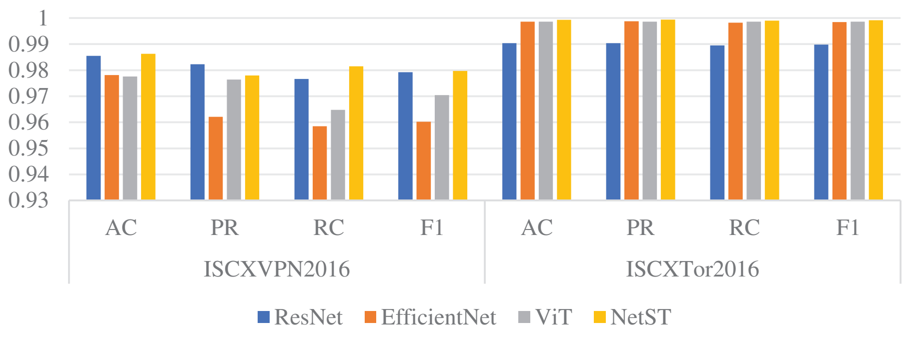


Figure 8: Comparison of evaluation indicators of different models

Table 8: Comparison of different optimizers and schedulers

Optimizer	**Scheduler**	**AC**				
		Cross Platform-Android	**Cross Platform-iOS**	**USTC-TFC2016**	**ISCXVPN2016**	**ISCXTor2016**
Adam	ReduceLROnPlateau	0.9774	0.9769	0.9995	0.9863	0.9993
	MultiStepLR	0.9721	0.9741	0.9987	0.9769	0.9992
	CosineAnnealingLR	0.9590	0.9703	0.9976	0.9783	0.9990
AdamW	ReduceLROnPlateau	0.9755	0.9750	0.9991	0.9805	0.9987
	MultiStepLR	0.9693	0.9662	0.9988	0.9762	0.9979
	CosineAnnealingLR	0.9723	0.9710	0.9958	0.9574	0.9963

4.4 Model Performance Analysis

In this section, we analyze model performance using ISCXVPN2016, a representative dataset for network encrypted traffic classification. Table 9 presents a performance comparison of four commonly used deep learning models: ResNet, EfficientNet, ViT, and our proposed model, NetST. The comparison focuses on the number of parameters (Params), floating-point operations (FLOPs), and throughput (FPS), highlighting significant differences in computational efficiency and inference speed among the models. Among these, the ResNet101 model has the highest number of parameters, totaling 42.51 million, which contributes to its relatively high computational complexity. Its floating-point operation count is 3.95 GMac. In contrast, the EfficientNet model has 22.61 million parSameters, which is lower than that of ResNet101, and its computational demand is only 0.14 GMac, significantly less than that of the other models, resulting in higher computational efficiency. EfficientNet achieves a throughput of 2159.34 images per second, surpassing that of ResNet101. The ViT model has 37.84 million parameters, which is close to ResNet101, but its computational demand is considerably higher at 15.16 GMac, far exceeding that of the other models. Due to this extremely high computational requirement, the throughput of ViT is only 289.56 images per second, the lowest among all models. This indicates a slower inference speed in practical applications, making it more suitable for scenarios with ample computational resources and less stringent inference time requirements. The NetST model, with a parameter count of 14.36 million and a computational demand of 1.41 GMac, achieves a maximum throughput of 2704.45 images per second. It demonstrates strong performance with the smallest number of parameters and low computational demand, indicating that the model can operate efficiently on resource-constrained devices while maintaining superior inference speed and performance.

Table 9: Comparison of Params, FLOPs, FPS of different models

Method	Params (M)	FLOPs (GMac)	FPS—Throughput (image/s)
ResNet101	42.51	3.95	1597.05
EfficientNet	22.61	0.14	2159.34
ViT	37.84	15.16	289.56
NetST(ours)	14.36	1.41	2704.45

As illustrated in Fig. 9, the four models—NetST, EfficientNet, ResNet101, and ViT—demonstrate varying throughputs (measured in frames per second, FPS) during the training process. NetST and EfficientNet maintain high throughputs throughout the training, stabilizing at approximately 2500 FPS and 2000 FPS, respectively. In contrast, ResNet101 exhibits a relatively lower throughput, yet still achieves a commendable level of around 1500 FPS. ViT, however, shows the lowest throughput, with only about 200 FPS, highlighting its disadvantage in computational efficiency. In summary, the NetST model strikes a better balance between computational efficiency and throughput, making it suitable for application scenarios that demand efficient reasoning while operating within limited computational resources.

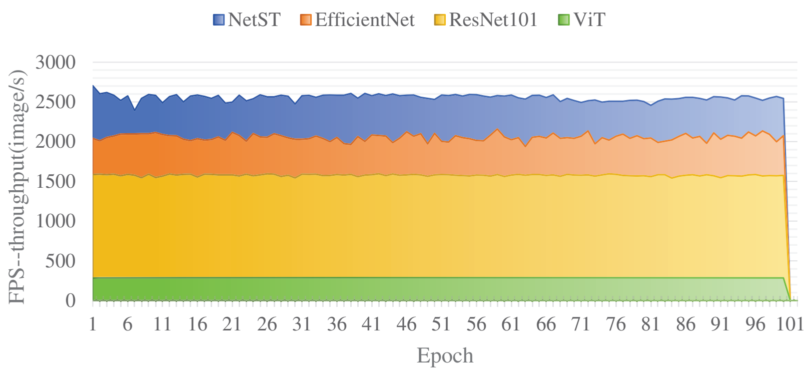


Figure 9: Comparison of the throughput of the different models

5 Conclusion

In this paper, we propose a method for classifying encrypted network traffic that integrates the Swin Transformer with a multilevel flow matrix representation. This approach addresses the complexity of network traffic data by comprehensively capturing traffic features through the multilevel flow matrix while leveraging the Swin Transformer's robust feature extraction capabilities to significantly enhance classification performance. Experimental results on multiple public datasets demonstrate that our method outperforms traditional machine learning and deep learning techniques in terms of accuracy, recall, and F1 score, exhibiting high classification accuracy and strong adaptability. Meanwhile, the NetST method offers significant advantages in terms of the number of parameters, computational efficiency, and throughput. Specifically, it utilizes only 37.9% of the parameters of the ViT, requires just 9.3% of the computational resources, and achieves a throughput that is 9.3 times greater than that of ViT. This makes it highly efficient and lightweight, making it suitable for scenarios with stringent resource and efficiency requirements in practical applications. However, with the ongoing evolution of HTTP/3 and QUIC-based multipath protocols, it is challenging to effectively capture their dynamic characteristics using fixed-size flow matrices and static model structures. In the future, we will integrate unsupervised learning techniques to enhance the feature representation of encrypted traffic by leveraging potential patterns in unlabeled data. Additionally, we will introduce a lightweight online incremental learning mechanism that allows the model to continuously adapt to new protocols and traffic distributions post-deployment, thereby further improving the performance and reliability of network encrypted traffic classification.

Acknowledgement: We would like to express our heartfelt gratitude to all those who contributed to this paper. Their dedication and insights were crucial in shaping the outcomes of this work.

Funding Statement: This research is partially supported by National Natural Science Foundation of China (62473341), Key Technologies R&D Program of Henan Province (242102211071, 252102211086, 252102210166).

Author Contributions: The authors confirm contribution to the paper as follows: Jianwei Zhang, study conception and design; Hongying Zhao, draft manuscript preparation; Yuan Feng, Zengyu Cai, data collection; Liang Zhu, analysis and interpretation of results. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data and materials utilized in this review originate from publicly available databases and previously published studies, with proper citations included throughout the text. References to these sources can be found in the bibliography.

Ethics Approval: Not applicable.

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

References

1. Liu C, He L, Xiong G, Cao Z, Li Z. FS-Net: a flow sequence network for encrypted traffic classification. In: IEEE INFOCOM 2019—IEEE Conference on Computer Communications; 2019 Apr 29–May 2; Paris, France. p. 1171–9. doi:10.1109/INFOCOM.2019.8737507.
2. Amiri Z, Heidari A, Navimipour NJ, Esmaeilpour M, Yazdani Y. The deep learning applications in IoT-based bio- and medical informatics: a systematic literature review. Neural Comput Appl. 2024;36(11):5757–97. doi:10.1007/s00521-023-09366-3.
3. Pang L, Gu W, Cao X. TRQ3DNet: a 3D quasi-recurrent and transformer based network for hyperspectral image denoising. Remote Sens. 2022;14(18):4598. doi:10.3390/rs14184598.
4. Xu K, Li W, Wang X, Hu X, Yan K, Wang X, et al. CUR transformer: a convolutional unbiased regional transformer for image denoising. ACM Trans Multimedia Comput Commun Appl. 2023;19(3):1–22. doi:10.1145/3566125.

5. Xue T, Ma P. TC-net: transformer combined with CNN for image denoising. Appl Intell. 2023;53(6):6753–62. doi:10.1007/s10489-022-03785-w.
6. Rafiepour M, Sartakhti JS. CTRAN: CNN-transformer-based network for natural language understanding. Eng Appl Artif Intell. 2023;126(C):107013. doi:10.1016/j.engappai.2023.107013.
7. Zhang H, Shafiq MO. Survey of transformers and towards ensemble learning using transformers for natural language processing. J Big Data. 2024;11(1):25. doi:10.1186/s40537-023-00842-0.
8. Okonkwo Z, Foo E, Li Q, Hou Z. A CNN based encrypted network traffic classifier. In: Proceedings of the 2022 Australasian Computer Science Week; 2022 Feb 14–18; Brisbane, Australia. p. 74–83. doi:10.1145/3511616.3513101.
9. Davis RE, Xu J, Roy K. Classifying malware traffic using images and deep convolutional neural network. IEEE Access. 2024;12:58031–8. doi:10.1109/access.2024.3391022.
10. Zhao R, Zhan M, Deng X, Wang Y, Wang Y, Gui G, et al. Yet another traffic classifier: a masked autoencoder based traffic transformer with multi-level flow representation. Proc AAAI Conf Artif Intell. 2023;37(4):5420–7. doi:10.1609/aaai.v37i4.25674.
11. Lin P, Ye K, Hu Y, Lin Y, Xu CZ. A novel multimodal deep learning framework for encrypted traffic classification. IEEE/ACM Trans Netw. 2023;31(3):1369–84. doi:10.1109/TNET.2022.3215507.
12. 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; 2022 Apr 25–29; Lyon, France. p. 633–42. doi:10.1145/3485447.3512217.
13. Wang K, Gao J, Lei X. MTC: a multi-task model for encrypted network traffic classification based on transformer and 1D-CNN. Intell Autom Soft Comput. 2023;37(1):619–38. doi:10.32604/iasc.2023.036701.
14. Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, et al. Swin Transformer: hierarchical vision transformer using shifted windows. In: IEEE/CVF International Conference on Computer Vision (ICCV); 2021 Oct 10–17; Montreal, QC, Canada. p. 9992–10002. doi:10.1109/ICCV48922.2021.00986.
15. Roy S, Shapira T, Shavitt Y. Fast and lean encrypted Internet traffic classification. Comput Commun. 2022;186:166–73. doi:10.1016/j.comcom.2022.02.003.
16. Ma X, Zhu W, Wei J, Jin Y, Gu D, Wang R. EETC: an extended encrypted traffic classification algorithm based on variant ResNet network. Comput Secur. 2023;128:103175. doi:10.1016/j.cose.2023.103175.
17. Izadi S, Ahmadi M, Nikbazm R. Network traffic classification using convolutional neural network and ant-lion optimization. Comput Electr Eng. 2022;101:108024. doi:10.1016/j.compeleceng.2022.108024.
18. 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); 2019 Apr 29–May 2; Paris, France. p. 680–7. doi:10.1109/INFCOMW.2019.8845315.
19. Zhang H, Xiao X, Yu L, Li Q, Ling Z, Zhang Y, et al. One train for two tasks: an encrypted traffic classification framework using supervised contrastive learning. arXiv:2402.07501. 2024.
20. Huang H, Lu Y, Zhou S, Zhang X, Li Z. CoTNeT: contextual transformer network for encrypted traffic classification. Egypt Inform J. 2024;26:100475. doi:10.1016/j.eij.2024.100475.
21. Zou A, Yang W, Tang C, Lu J, Guo J. A novel and effective encrypted traffic classification method based on channel attention and deformable convolution. Comput Electr Eng. 2024;118:109406. doi:10.1016/j.compeleceng.2024.109406.
22. Okonkwo Z, Foo E, Hou Z, Li Q, Jadidi Z. Encrypted network traffic classification with higher order graph neural network. In: Australasian Conference on Information Security and Privacy; 2023 Jul 5–7; Brisbane, QLD, Australia. p. 630–50. doi:10.1007/978-3-031-35486-1_27.
23. Zhang L, Tan JW, Man DP, Han S, Ma S. Research on mobile application classification technology for unidirectional encrypted traffic. J Integr Technol. 2024;13(5):40–52.
24. Sun ZP, Wang ZH, Pan W. Homogeneous traffic identification method for encrypted mobile applications based on graph attention mechanism. J Cybersecur. 2024;2(2):97–106. (In Chinese). doi:10.20172/j.issn.2097-3136.240210.
25. Cheng G, Wang S. Traffic classification based on port connection pattern. In: 2011 International Conference on Computer Science and Service System (CSSS); 2011 Jun 27–29; Nanjing, China: IEEE; 2011. p. 914–7. doi:10.1109/CSSS.2011.5974374.

26. Kong L, Huang G, Wu K. Identification of abnormal network traffic using support vector machine. In: 2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT); 2017 Dec 18–20; Taipei, Taiwan. p. 288–92. doi:10.1109/PDCAT.2017.00054.
27. He H. A network traffic classification method using support vector machine with feature weighted-degree. J Digit Inf Manag. 2017;15(2):76–83.
28. Chen L, Gao S, Liu B, Lu Z, Jiang Z. THS-IDPC: a three-stage hierarchical sampling method based on improved density peaks clustering algorithm for encrypted malicious traffic detection. J Supercomput. 2020;76(9):7489–518. doi:10.1007/s11227-020-03372-1.
29. Berkay Celik Z, Walls RJ, McDaniel P, Swami A. Malware traffic detection using tamper resistant features. In: MILCOM 2015—2015 IEEE Military Communications Conference; 2015 Oct 26–28; Tampa, FL, USA. p. 330–5. doi:10.1109/MILCOM.2015.7357464.
30. 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); 2017 Jan 11–13; Da Nang, Vietnam. p. 712–7. doi:10.1109/ICOIN.2017.7899588.
31. 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); 2017 Jul 22–24; Beijing, China. p. 43–8. doi:10.1109/ISI.2017.8004872.
32. Wang H, Xu T, Yang J, Wu L, Yang L. Sessionvideo: a novel approach for encrypted traffic classification via 3D-CNN model. In: 2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS); 2020 Sep 28–30; Takamatsu, Japan.
33. Wang X, Chen S, Su J. App-net: a hybrid neural network for encrypted mobile traffic classification. In: IEEE INFOCOM 2020—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); 2020 Jul 6–9; Toronto, ON, Canada. p. 424–9. doi:10.1109/infocomwkshps50562.2020.9162891.
34. Kong X, Wang C, Li Y, Hou J, Jiang T, Liu Z. Traffic classification based on CNN-LSTM hybrid network. In: 18th International Forum on Digital TV and Wireless Multimedia Communications; 2021 Dec 3–4; Shanghai, China. p. 401–11. doi:10.1007/978-981-19-2266-4_31.
35. Liu K, Zhang Y, Zhang X, Qiao W, Dong P. Network traffic classification based on LSTM + CNN and attention mechanism. In: International Conference on Emerging Networking Architecture and Technologies; 2022 Nov 15–17; Shenzhen, China. p. 545–56. doi:10.1007/978-981-19-9697-9_44.
36. Wu Z, Long Z. Research on network traffic classification method based on CNN–RNN. In: 3D imaging—multidimensional signal processing and deep learning. Singapore: Springer Nature; 2023. p. 249–57. doi:10.1007/978-981-99-1145-5_24.
37. Zhao H, Sun S, Wang R, Fan F, Liu L. CTCRNN: a hybrid CNN-BiLSTM based classification method for industrial IoT encrypted traffic based on spatiotemporal features. In: Proceedings of the 2024 3rd International Conference on Cyber Security, Artificial Intelligence and Digital Economy; 2024 Mar 1–3; Nanjing, China. doi:10.1145/3672919.3672920.
38. Lotfollahi M, Jafari Siavoshani M, Shirali Hossein Zade R, Saberian M. Deep packet: a novel approach for encrypted traffic classification using deep learning. Soft Comput. 2020;24(3):1999–2012. doi:10.1007/s00500-019-04030-2.
39. He Y, Li W. Image-based encrypted traffic classification with convolution neural networks. In: 2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC); 2020 Jul 27–30; Hong Kong, China. p. 271–8. doi:10.1109/dsc50466.2020.00048.