Yet Another Traffic Classifier: A Masked Autoencoder Based Traffic Transformer with Multi-Level Flow Representation

Ruijie Zhao[1*], **Mingwei Zhan**[1*], **Xianwen Deng**[1], **Yanhao Wang**[2], **Yijun Wang**[1], **Guan Gui**[3†], **Zhi Xue**[1†]

[1]Shanghai Jiao Tong University, Shanghai, China
[2]QI-ANXIN, Beijing, China
[3]NJUPT, Nanjing, China
{ruijiezhao, mw.zhan, 2594306528, ericwyj, zxue}@sjtu.edu.cn; wangyanhao136@gmail.com; guiguan@njupt.edu.cn

Abstract

Traffic classification is a critical task in network security and management. Recent research has demonstrated the effectiveness of the deep learning-based traffic classification method. However, the following limitations remain: (1) the traffic representation is simply generated from raw packet bytes, resulting in the absence of important information; (2) the model structure of directly applying deep learning algorithms does not take traffic characteristics into account; and (3) scenario-specific classifier training usually requires a labor-intensive and time-consuming process to label data. In this paper, we introduce a masked autoencoder (MAE) based traffic transformer with multi-level flow representation to tackle these problems. To model raw traffic data, we design a formatted traffic representation matrix with hierarchical flow information. After that, we develop an efficient Traffic Transformer, in which packet-level and flow-level attention mechanisms implement more efficient feature extraction with lower complexity. At last, we utilize the MAE paradigm to pre-train our classifier with a large amount of unlabeled data, and perform fine-tuning with a few labeled data for a series of traffic classification tasks. Experiment findings reveal that our method outperforms state-of-the-art methods on five real-world traffic datasets by a large margin. The code is available at https://github.com/NSSL-SJTU/YaTC.

1 Introduction

Traffic classification is attracting the attention of service providers and equipment vendors as a significant solution for solving network management problems. Understanding the content of traffic allows network operators to respond swiftly in support of diverse business goals, enhancing service quality and user experience (Papadogiannaki and Ioannidis 2021). Meanwhile, traffic classification is a critical component of the intrusion detection system, which is utilized to detect threats and safeguard system security (Yao et al. 2019). However, the growing usage of encrypted traffic and anonymous network technology makes analyzing complicated traffic difficult. To construct a sophisticated traffic analyzer for more accurate traffic classification, it is necessary to capture implicit and robust patterns in diverse traffic.

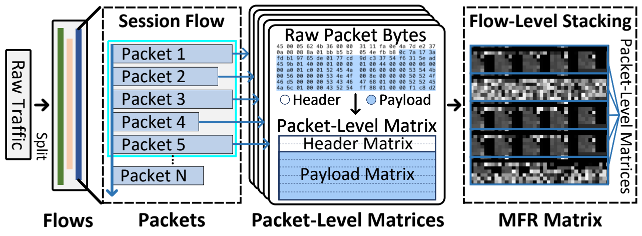

Figure 1: The schematic illustration of multi-level flow representation (MFR).

Traditional traffic analysis methods identify different network services based on fundamental traffic features (e.g., communication protocol, port number), which are no longer appropriate owing to the complexity and variability of the current traffic (Taylor et al. 2016). To address this issue, several works use machine learning (ML) algorithms to classify traffic through statistical features. However, these approaches rely on expert experience to select specific features, and even insignificant statistical features might impact analysis performance (Shen et al. 2020). Deep learning (DL) techniques, which can automatically extract features from packet bytes of raw traffic, have been used by an increasing number of researchers for more effective traffic analysis in recent years. Unfortunately, once the length of a certain packet is too long, the bytes from that packet can overwhelm important information from other packets. Thus, the lack of representation design in traditional DL-based methods brings instability in performance. Besides, they often need a large amount of labeled traffic data for training. Labeling data is a time-consuming and labor-intensive operation, resulting in high deployment and update costs.

Pre-training methods with massive volumes of unlabeled data have recently demonstrated outstanding performance in computer vision (CV) (He et al. 2022; Caron et al. 2021; Chen, Xie, and He 2021) and natural language processing (NLP) (Devlin et al. 2019; Lan et al. 2020) tasks. When compared to the traditional supervised learning paradigm, the self-supervised learning paradigm pre-training method

[*]These authors contributed equally.
[†]Corresponding authors.

has two major advantages: (1) more effective representations can reduce the reliance of downstream tasks on the amount of labeled data, and (2) better initialization parameters can speed up convergence and improve classification performance. Recent studies (He, Yang, and Chen 2020; Lin et al. 2022) in traffic classification directly utilize the BERT-based paradigm (Devlin et al. 2019), a pre-training method created for NLP tasks, and obtain better performance. However, unlike typical words, traffic bytes as input lack explicit high-level semantic units, making it unreasonable to build vocabularies to learn semantic relations. Furthermore, neither method designs a dedicated model structure for traffic classification based on traffic characteristics.

To address the challenges mentioned above, we introduce Yet Another Traffic Classifier (YaTC), a novel traffic classifier with self-supervised learning that learns latent representations from large amounts of unlabeled traffic data for more successful classification. Specifically, we first create a multi-level flow representation (MFR) matrix to model the raw traffic. It is generated from raw packet bytes and contains traffic information at different granularities through a formatted matrix, as illustrated in Figure 1. Then, a novel Traffic Transformer with a packet-level attention module and flow-level attention module is constructed to analyze traffic using the MFR matrix. Finally, we train our classifier based on the masked autoencoder (MAE) paradigm in two stages: pre-training and fine-tuning. In the pre-training stage, we randomly mask some packet bytes in the MFR matrix as input. Next, an autoencoder-based network, which consists of a traffic encoder and a small-scale decoder, reconstructs the original MFR matrix. Our traffic encoder leverages a large amount of unlabeled traffic data to learn latent representations through the reconstruction task. In the fine-tuning stage, we load the parameters of the pre-trained traffic encoder into Traffic Transformer and fine-tune it with a small number of labeled data for traffic classification. Our contributions can be briefly summarized as follows:

- We propose an MAE-based Traffic Transformer with MFR, called YaTC, for traffic classification. YaTC breaks through the traditional traffic analysis methods in three perspectives: traffic representation, classifier structure, and training strategy.
- We design an MFR matrix that fully considers the flow hierarchy to represent the raw traffic. To effectively utilize the MFR matrix for traffic analysis, we build a novel Traffic Transformer with packet-level and flow-level attention mechanisms. It can perform more efficient feature extraction with lower complexity and fewer parameters.
- We apply the MAE-based self-supervised learning paradigm to train our classifier. It first leverages large-scale unlabeled traffic data to learn generic latent representations, and then performs fine-tuning with a small amount of labeled data for a series of traffic classification tasks.
- We evaluate YaTC on five real-world traffic datasets. Experimental results show that our method outperforms the state-of-the-art methods by a large margin.

2 Related Work

2.1 Traffic Analysis Methods

Rule-Based Methods. In the early stage, researchers mainly used rule-based methods to accomplish traffic analysis. Through rules designed by security experts, basic attributes such as communication protocol and port number of traffic data are used to discover behaviors that violate security policies (Nguyen and Armitage 2008). However, as network services and environments become more complicated, the fundamental features are insufficient to fulfill the current traffic analysis requirement (Taylor et al. 2016).

ML-Based Methods. To analyze complicated traffic effectively, ML algorithms are introduced to explore the high-dimensional statistical features of traffic. For instance, CUMUL (Panchenko et al. 2016) selects 104 optimal statistical features through accuracy evaluation, which are then utilized as input for a support vector machine (SVM) to identify website traffic; AppScanner (Taylor et al. 2016) adopts the random forest (RF) to analyze the statistical features generated by the traffic of different applications; isAnon (Cai et al. 2019) designs a hybrid feature selection algorithm to filter out redundant statistical features, and uses the extreme gradient boosting algorithm to detect anonymity network traffic. Although ML-based methods combined with statistical features can analyze complex traffic, they rely on statistical features designed by experts and need to select optimal features for different scenarios (Shen et al. 2020).

DL-Based Methods. DL-based approaches, which analyze traffic based on raw packets rather than human-designed features, have become the primary tool to automatically extract traffic representations and achieve remarkable performance improvement. In (Wang et al. 2017), a convolutional neural network (CNN) is used to identify traffic images for encrypted traffic classification. (Zhang et al. 2020) adopts the improved CNN classifier that leverages multiple channels to enrich feature information and improve analysis efficiency. TSCRNN (Lin, Xu, and Gao 2021) combines CNN and recurrent neural network (RNN) to learn the temporal characteristics from time-related packets. However, some existing DL-based methods directly use the packet bytes in the flow to represent the raw traffic, resulting in the bytes of a long packet overwhelming important information from other packets. Besides, they rely on sufficient labeled training data, and it is intractable to collect and manually label sufficient real traffic samples. Thus, we design an MFR matrix to represent multi-level information of the raw traffic through a formatted matrix and develop a novel Traffic Transformer to implement more efficient feature extraction based on these levels. To reduce the dependence on labeled data, we first leverage large-scale unlabeled data for pre-training, and then fine-tune with a few labeled data for traffic classification.

2.2 Pre-training Methods

Pre-training methods significantly reduce the appetite for labeled training data by self-supervised learning. Besides, unbiased data representations learned from large amounts

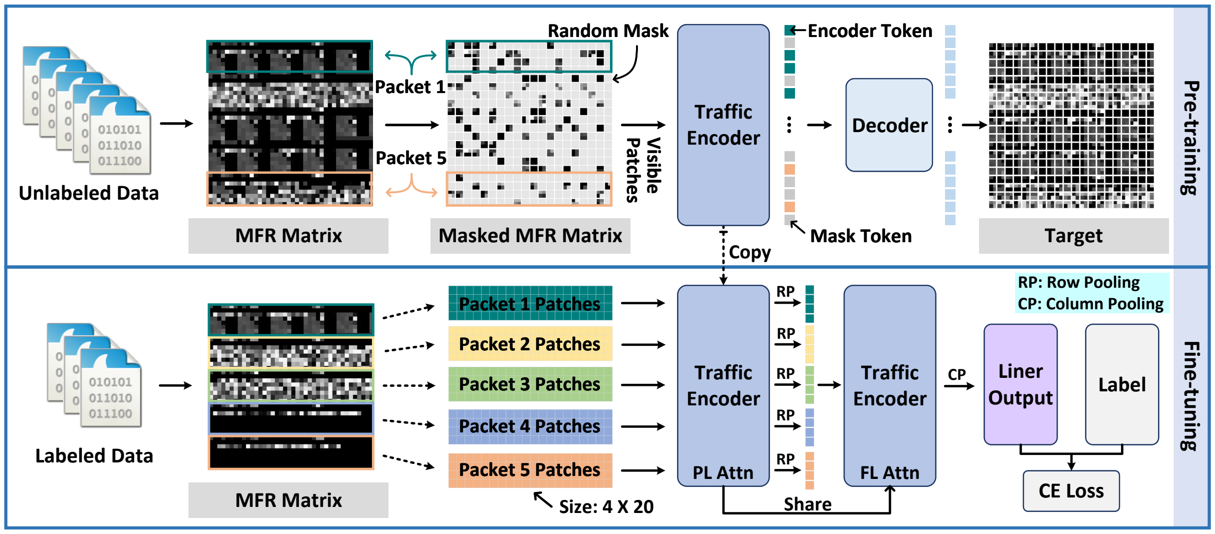

Figure 2: The schematic illustration of YaTC.

of unlabeled data further improve performance on downstream tasks. Overall, the pre-training methods first obtain the pre-trained model with unlabeled data, and then load the model parameters to complete downstream tasks. In NLP tasks, BERT (Devlin et al. 2019) is the most widely used to implement pre-trained models by cloze task and next sentence prediction. Some traffic classification studies (He, Yang, and Chen 2020; Lin et al. 2022) consider the bytes of traffic packets as words and introduce BERT for pre-training to achieve better classification performance. However, since the input traffic bytes lack explicit high-level semantic units, latent representation extraction of traffic bytes is more suitable as a CV task than an NLP task. In CV tasks, traditional pre-training methods (Chen, Xie, and He 2021; Caron et al. 2021) rely on data augmentation (e.g., cropping, enlarging) to form pairs of positive and negative samples for pre-training via contrastive learning. Obviously, data augmentation on the traffic image generated by the packet bytes will seriously damage the original information, making it difficult to apply the contrastive learning paradigm to our task. Benefiting from the introduction of the MAE (He et al. 2022), we are able to obtain an effective pre-trained model by masking patches for reconstruction training. Since the masking operation does not change other fixed-position packet bytes in the traffic image, it conforms to the design of fixed positions of different levels of packet content in multi-level flow representation.

3 Methodology

In this section, we introduce our traffic representation method, classifier structure, and training strategy in detail in Section 3.1, Section 3.2, and Section 3.3, respectively. The overview of our YaTC is shown in Figure 2.

3.1 Multi-Level Flow Representation

We design a novel method to produce multi-level flow representations from raw traffic data as input for traffic classification. Most existing methods directly intercept the preceding fixed number of bytes in the flow to form a two-dimensional matrix, which can be handled as an image and classified by DL algorithms. However, compared with the header, the size of the payload is usually significantly larger and is full of incomprehensible information generated by encrypt operation, resulting in excessive low-level semantic information in the matrix, which affects these models' effectiveness and efficiency. Moreover, in some flows, the first long packet will occupy the entire matrix, so the matrix cannot include the data in other packets of the traffic flow.

To address the above issues, we propose an MFR matrix with a formatted two-dimensional matrix to represent raw traffic flow, as illustrated in Figure 1. First, we split the raw traffic into flows according to IP address, port number, and protocol type. Then, to avoid introducing biased interference, we remove the Ethernet headers of the flows, set the port numbers to zero, and replace the IPs with random addresses but keep their directions. Finally, the M adjacent packets in the flow are captured and formatted into a two-dimensional matrix of size $H * W$ as the representation of this flow. We still use the raw bytes as the initial characteristics of the traffic, i.e., the value of each point in the traffic representation matrix, but with the following particular design for capturing the multi-level information of the raw traffic.

- **Byte-level.** Each row of the traffic representation matrix includes only one type of traffic bytes, classified into header row and payload row.
- **Packet-level.** Each packet is represented by the header matrix and payload matrix, forming a packet-level matrix of size $H/M * W$.
- **Flow-level.** Since the flow is composed of ordered packets, M adjacent packet-level matrices are stacked in the second dimension to form the final MFR matrix.

In this way, the information at each level is fixed, and there will be no overflow at the lower level leading to the loss of information at the higher level. We set our MFR matrix to include 5 packet-level matrices with a total of 40 rows, and each row includes 40 bytes. Each packet's header is represented by 2 header rows with 80 bytes, with the capability to contain the IP layer header (20 bytes), TCP header (20 bytes) or UDP header (8 bytes), and optional headers. We assign 6 payload rows to accommodate the payload of each packet and perform interception when it exceeds 240 bytes. Note that if the number of valid bytes is insufficient, it will be padded with 0 bytes to form a fixed-size representation matrix.

3.2 Traffic Transformer

According to the characteristics of traffic, we design our Traffic Transformer composed of an embedding module, a packet-level attention module, and a flow-level attention module, to effectively conduct traffic classification with the MFR matrix.

Embedding Module. The MFR matrix $x \in \mathcal{R}^{H \times W}$ is first split into non-overlapping 2D patches of size $P \times P$ with a total number of $N = HW/P^2$, denoted as $x_p \in \mathcal{R}^{N \times P^2}$. The patches are then mapped to D-dimensional vectors by a linear layer as patch embeddings. To maintain the position information, which is crucial in MFR, the position embeddings are added to the patch embeddings as the input of the traffic encoder.

$$x_0 = [x_p^1 E; x_p^2 E; ...; x_p^N E] + E_{pos}. \tag{1}$$

Specifically, we take $D = 192$, $P = 2$ and $N = 20 * 20 = 400$, so that all elements of one patch represent the same type of raw bytes, and each packet matrix contains one row of header patches and three rows of payload patches.

Packet-level Attention Module. In the packet-level attention phase, we develop a traffic encoder based on Vision Transformers (Dosovitskiy et al. 2021), which consists of alternating multi-head self-attention layers (MSA) and feed-forward layers.

Aiming to preferentially learn the dependencies between header patches or payload patches within the packet, the traffic encoder only performs multi-headed self-attention among patches in the same packet rather than all patches in the MFR matrix. The MSA layer allows patches of the packet to interact with each other effectively according to the degree of relevance by different heads $Concat(head_1, \cdots, head_n)$, and each head is computed by the attention function:

$$Q = x_l W^Q, K = x_l W^K, V = x_l W^V, \tag{2}$$

$$Attn(Q, K, V) = softmax(\frac{QK^T}{\sqrt{D_k}})V, \tag{3}$$

where $W^Q, W^K, W^V \in \mathcal{R}^{D \times D_k}$ are learnable parameters. In this work, we employ $n = 16$ parallel attention heads and $L = 4$ alternating layers.

Note that our packet-level attention module focuses on promoting packet-level information interaction at this stage, which is also in line with the traffic characteristics that the information in the packet has a stronger correlation. In other words, it ignores the patch features of other packets and outputs a valid packet-level representation. Furthermore, our method also has lower time complexity compared to global attention ($O(N^2/M)$ vs. $O(N^2)$), which can benefit the efficiency of traffic analysis.

Flow-level Attention Module. After packet-level attention, significant packet-level features of each patch of the MFR matrix $x'_p \in \mathcal{R}^{N \times D}$ have been output, which provides the basis for further use of the transformer to extract long-distance relational dependencies between different packets on the flow. At the flow level, it is unnecessary to continue with the same fine-grained patch as before, and even simply mean-pooling all the packet-level patch features can yield good performance. To learn inter-packet relationships at a coarser granularity, we employ row pooling (RP), which partially mean-pooling the output patches' features of the packet-level attention by row to generate row patches:

$$x_r = Pooling(x'_p), \tag{4}$$

where $x_r \in \mathcal{R}^{\sqrt{N} \times D}$ are row patches output of RP. Since each row of input patch features indicates the same type byte of traffic, the outputs of RP also can be divided into header or payload with fixed meaning.

In particular, each packet contains 1 header row patch and 3 payload row patches, representing the partial features within the packet and a MFR matrix has a total of $\sqrt{N}$ row patches. We input all the row patches within an MFR matrix into a traffic encoder, and the MSA layers capture the flow-level information between the row patches. The final output is a column of row patch features $x_c \in \mathcal{R}^{\sqrt{N} \times D}$, which are further performed column pooling (CP) to obtain the final representation $x_{MFR} \in \mathcal{R}^D$ of the entire MFR matrix:

$$x_{MFR} = Pooling(x_c). \tag{5}$$

Besides, with the coarser granularity, the model also has a lighter weight, the number of row patches is only $\sqrt{N}$ and the time complexity of this stage is only $O(N)$.

3.3 Training Strategy of YaTC

Pre-training YaTC. As shown in Figure 2, in the pre-training stage, YaTC utilizes MAE (He et al. 2022) paradigm with an asymmetric encoder-decoder architecture to reconstruct the raw bytes in the MFR matrix. A high percentage of the MFR patches are randomly masked, and only a tiny fraction of the patches (i.e. visible unmasked patches) are input to the model. After that, our traffic encoder extracts as many valid features of this fraction of the patches as possible and then outputs the encoder tokens. Finally, a small decoder uses encoder tokens and mask tokens to recover the masked region of the MFR matrix. MAE is trained with a reconstruction loss, i.e., mean squared error (MSE) between the ground-truth pixels y_{real} and the predicted pixels y_{rec}:

$$\mathcal{L}_{rec} = MSE(y_{rec}, y_{real}). \tag{6}$$

After pre-training, the traffic encoder can extract high-quality features and is preserved for downstream tasks.

The pre-training does not require costly manual labeling, allowing the use of large amounts of unlabeled data from authentic scenes. Moreover, we find that a very high mask ratio

(90%) achieves better results on downstream tasks. A high mask ratio also means that only a few patches are visible, resulting in lacking raw information to capture dependencies within and between packets. Thus, we perform global attention instead of packet-level attention and flow-level attention during pre-training. The difficult pre-training task and global attention would enable the traffic encoder to capture features on packet-level and flow-level together with minimal information.

Fine-tuning YaTC. On the downstream task, the encoder parameters from pre-training are loaded into both the packet-level attention module and flow-level attention module in the traffic transformer and used for feature extraction on packet-level and flow-level. For classification, the features of each patch are mean-pooled in two stages (RP and CP) to be used as the classification features of the MFR Matrix, which are flattened and input to an MLP to obtain the prediction distribution $\hat{y} \in \mathcal{R}^C$, where C is the number of traffic categories. Then classification loss is computed according to cross-entropy loss between prediction distribution $\hat{y}$ and the ground-truth label y:

$$\mathcal{L}_{CE} = H(\hat{y}, y). \tag{7}$$

To speed up convergence and reduce the model size, inspired by ALBERT (Lan et al. 2020) and CYCLE (Takase and Kiyono 2021), we shared parameters between the two traffic encoders and achieved better performance with half the number of parameters. We argue the reason that both packet-level attention and flow-level attention are essentially performing inter-patch dependency capture, and the information interactions during multi-headed self-attention are similar. Moreover, the data in the same row of patches belong to the same type, i.e., header or payload, and the row-pooling's mean operation does not change the feature space in which the patches are located. So except for the difference in granularity, the two traffic encoders on the feature space are very similar to the stacking of transformer layers, which provides the conditions for the application of parameter sharing. In the fine-tuning stage, the two traffic encoders cause the double depth of Transformer layers compared to the pre-training stage, but the amount of training data is very limited. Thus, parameter sharing also makes the model easier to train.

4 Experiments

4.1 Experiment Settings

Data Preparation. Our experiments are conducted on five public real-world encrypted traffic datasets ISCXVPN2016 dataset (Habibi Lashkari et al. 2016), ISCXTor2016 dataset (Lashkari et al. 2017), USTC-TFC2016 dataset (Wang et al. 2017), CICIoT2022 dataset (Dadkhah et al. 2022), and Cross-Platform dataset (Van Ede et al. 2020), respectively.

To verify the generality of the method, we conduct evaluations on the above five datasets. The first four training datasets form a large-scale unlabeled training dataset for pre-training. In the fine-tuning stage, our classifier uses five training datasets for supervised learning to complete the corresponding traffic classification task. Note that a generic pre-trained model is obtained in the pre-training stage, while the classifier training in the fine-tuning stage is performed separately. Furthermore, since the training dataset of Cross-Platform is not part of the unlabeled dataset in the pre-training stage, the Cross-Platform dataset is used for transfer learning experiments in Section 4.5.

Implementation Details. In the pre-training phase, the batch size is 512 and the total step is 150,000. We use the linear learning rate scaling rule and the base learning rate is set to 1×10^{-3} with the AdamW optimizer. The masking ratio for randomly masking patches is set to 0.9. Then, we use the AdamW optimizer in fine-tuning for 200 epochs, where the base learning rate is set to 2×10^{-3} and batch size is 64. The impact of different masking ratios is discussed in Section 4.5. The proposed method is implemented using PyTorch 1.9.0 and trained on a server with four NVIDIA GeForce RTX3090 GPUs.

Evaluation Metrics. To measure the classification performance of our method, we calculate the number of True Positive (T_p), True Negative (T_n), False Positive (F_p), and False Negative (F_n). Based on the above definition, Recall, Precision, and F_1 can be obtained:

$$Recall = \frac{T_p}{T_p + F_n}, \quad Precision = \frac{T_p}{T_p + F_p}, \tag{8}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{9}$$

4.2 Comparison with State-of-the-Art Methods

To comprehensively evaluate our method, we compare YaTC with a range of baselines and state-of-the-art methods, as listed below:

- **FlowPrint** (Van Ede et al. 2020) and **AppScanner** (Taylor et al. 2016) are ML-based methods using statistical features for traffic classification.
- **DF** (Sirinam et al. 2018), **Deeppacket** (Lotfollahi et al. 2020), **2D-CNN** (Wang et al. 2017), **3D-CNN** (Zhang et al. 2020), and **FS-Net** (Liu et al. 2019) are DL-based traffic analysis methods, which use raw packet information for supervised learning.
- **PERT** (He, Yang, and Chen 2020) and **ET-BERT** (Lin et al. 2022) treat traffic representation extraction as an NLP task for pre-training, and then fine-tune the classifier with limited labeled data.

As is shown in Table 1, the results demonstrate the effectiveness of DL-based methods and the insufficiency of ML-based methods with statistical features. Since only the vector of packets' directions in the flow is used as input features, the classification performance of DF is significantly lower than other DL-based methods. Furthermore, we observe that methods without pre-training on the ISCXTor2016 dataset perform poorly. The reason is that encryption and obfuscation techniques for anonymous traffic make it difficult to analyze the payload directly. However, pre-training-based methods learn latent representations

Method	ISCXVPN2016		ISCXTor2016		USTC-TFC2016		CICIoT2022	
	Acc.	F_1	Acc.	F_1	Acc.	F_1	Acc.	F_1
FlowPrint	30.29%	14.09%	25.27%	10.19%	25.30%	12.47%	50.46%	49.14%
AppScanner	79.93%	80.85%	50.27%	49.68%	60.41%	58.36%	76.52%	76.81%
DF	62.87%	25.40%	33.24%	7.00%	58.45%	49.15%	60.13%	46.35%
Deeppacket	80.21%	80.17%	36.81%	26.81%	88.49%	88.83%	88.28%	88.08%
2D-CNN	81.26%	80.64%	34.62%	33.66%	92.26%	92.05%	90.07%	90.00%
3D-CNN	81.09%	80.79%	34.89%	33.96%	91.55%	91.16%	89.39%	89.33%
FS-Net	87.64%	87.30%	52.03%	51.64%	87.05%	86.02%	85.37%	85.30%
PERT	88.62%	88.61%	80.22%	79.99%	96.63%	96.64%	90.52%	90.49%
ET-BERT	87.74%	87.47%	65.38%	64.98%	96.95%	96.95%	90.35%	90.31%
Ours (YaTC)	**98.07%**	**98.04%**	**99.72%**	**99.72%**	**97.86%**	**97.86%**	**96.58%**	**96.58%**

Table 1: Comparison results on ISCXVPN2016, ISCXTor2016, USTC-TFC2016, and CICIoT2022 datasets.

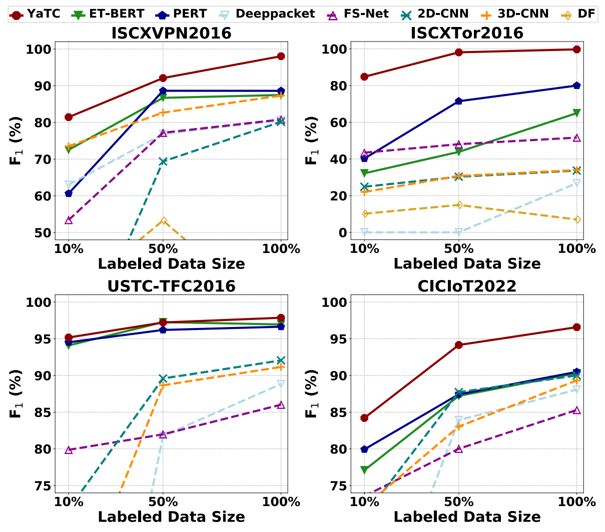

Figure 3: The performance comparison with other models using different labeled sample sizes.

from a large amount of unlabeled traffic data for more effective classification. Additionally, pre-training methods outperform other methods on all datasets except CICIoT2022, illustrating that pre-trained classifiers can achieve better performance on downstream traffic classification tasks. Benefiting from well-designed traffic representation, model structure, and training paradigm, YaTC consistently outperforms all compared methods by a large margin on all datasets. It can be concluded that our method is suitable for traffic analysis in various scenarios as well as has excellent performance.

4.3 Few-Shot Analysis

To validate the robustness of YaTC in few-shot scenarios, we set the labeled data size to 10%, 50%, and 100% and compare YaTC with other baselines and state-of-the-art methods on four datasets. Experimental results are shown in Figure 3. Experiments show that the three pre-training-based methods, YaTC, ET-BERT, and PERT, generally outperform other supervised methods in few-shot scenarios. Some other methods achieve similar performance to pre-training-based methods in few cases, but none of them could maintain it on all four datasets. The pre-training enables the model to master the ability to extract high-quality representations using unlabeled data, thereby greatly reducing the dependence on labeled data. Besides, YaTC achieves better performance than ET-BERET and PERT at all sizes of labeled data, indicating excellent robustness.

4.4 Ablation Study

To further examine the level of benefit that each component of YaTC brings to the performance, an ablation study is performed on four baseline datasets. The evaluation results are reported in Table 2. First, compared with applying global attention, our method not only reduces the complexity (discussed in Section 3.2) but also achieves better results, demonstrating the benefits of packet-level and flow-level attention mechanisms. Furthermore, removing packet-level attention leads to significant performance degradation under all configurations, which illustrates the considerable contribution of packet-level feature extraction. We note that flow-level attention without pre-training sometimes leads to performance loss, which would not occur after pre-training. This is because further attention to small data without pre-training is easier to cause over-fitting. The ablations also show that applying parameter sharing during fine-tuning brings both lightweight and performance enhancements. Moreover, directly transforming the raw traffic bytes without MFR or removing only the flow-level stacking in MFR also results in weaker performance, obviously for the encrypted traffic tasks.

4.5 Discussions

Impact of Masking Ratio. To explore the effect of the masking ratio on the quality of learned representation, we conduct ablation experiments on four datasets with a range of masking ratios. Experimental results are shown in Figure 4. Overall, a higher mask ratio will result in better performance. On the other hand, an excessively high masking ratio makes the reconstruction task too difficult, and the F1 score

Method	ISCXVPN2016		ISCXTor2016		USTC-TFC2016		CICIoT2022	
	Acc.	F_1	Acc.	F_1	Acc.	F_1	Acc.	F_1
Ours (YaTC)	**98.07%**	**98.04%**	**99.72%**	**99.72%**	**97.86%**	**97.86%**	**96.58%**	**96.58%**
Ours with GA	95.27%	95.14%	98.63%	98.62%	97.86%	97.86%	95.64%	95.61%
Ours w/o PA	90.19%	90.03%	78.02%	77.28%	96.03%	96.03%	92.84%	92.78%
Ours w/o FA	95.62%	95.49%	99.18%	99.18%	97.66%	97.62%	95.81%	95.80%
Ours w/o FS	92.47%	92.35%	97.80%	97.77%	93.48%	93.48%	94.58%	94.57%
Ours w/o PS	97.55%	97.53%	99.45%	99.45%	97.45%	97.40%	95.41%	95.39%
Ours w/o PT	87.74%	87.22%	92.03%	91.90%	95.32%	95.25%	92.70%	92.65%
Ours w/o PT & PA	78.63%	77.58%	39.84%	38.58%	93.28%	93.22%	90.88%	90.79%
Ours w/o PT & FA	87.74%	87.40%	85.99%	85.84%	95.52%	95.46%	93.19%	93.17%
Ours w/o PT & FS	81.96%	81.84%	83.52%	83.15%	91.75%	91.48%	91.59%	91.59%
Ours w/o PT & MFR	80.91%	80.49%	42.86%	42.11%	93.99%	93.90%	91.36%	91.26%

Table 2: Ablation study of key components in YaTC on ISCXVPN2016, ISCXTor2016, USTC-TFC2016, and CICIoT2022 datasets. The abbreviations are explained as follows, GA: global attention, PA: packet-level attention, FA: flow-level attention, FS: flow-level stacking, PS: parameter sharing, MFR: multi-level traffic representation, and PT: pre-training.

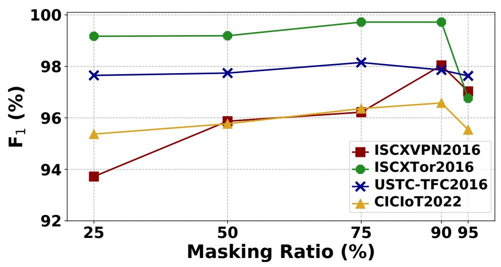

Figure 4: The impact of masking ratio on the quality of learned representation.

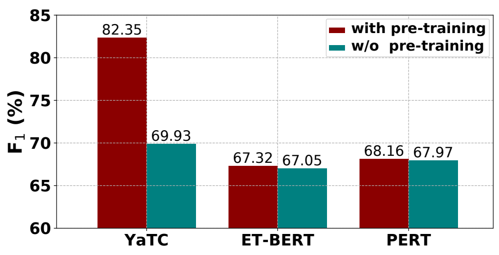

Figure 5: The performance comparison with the other two pre-training methods on the Cross-Platform dataset.

decreases rapidly. The F_1 score reaches the highest when the masking ratio is 90% on most datasets except USTC-TFC2016, which also has a high best masking ratio of 75%. This supports our view of the traffic task as being more similar to images than to words. In the masking strategy of NLP, such as BERT, the optimal masking radio maintains a low value and usually does not exceed 20%, which is the opposite of the CV according to the recent study (He et al. 2022). A high optimal masking rate implies a large information redundancy, and traffic classification does not require a thorough understanding of its content, which is also impossible on the encrypted payload. For classification tasks, words are high-level and abstract information, but traffic bytes are just sparse features without clear semantic units, similar to pixels. Hence, rather than the BERT-based methods treating raw bytes as words, our image-like MFR matrix is more rational.

Transfer Learning Experiments. We evaluate transfer learning on the Cross-Platform dataset. Two other pre-training traffic classification methods (PERT and ET-BERT) are introduced for comparison. The pre-trained models are all identical to those used in the previous experiments (i.e., pre-trained with the other four datasets). It can be seen from Figure 5 that our YaTC significantly improves the F_1 score from 69.93% to 82.35%, which is 12.42% better than without pre-training. Furthermore, both ET-BERT and PERT have weak boosts using pre-training, indicating that their pre-trained models are difficult to transfer to the new downstream traffic classification task.

5 Conclusion

In this work, we proposed YaTC, an MAE-based Traffic Transformer with MFR for traffic classification. With the MAE-based self-supervised learning paradigm, YaTC first learns generic latent representations from a large amount of unlabeled traffic data in the pre-training stage, then performs supervised learning with a few labeled data for a series of traffic classification tasks in the fine-tuning stage. Our method is evaluated on five real-world traffic datasets. Experimental results show that our YaTC is ahead of the state-of-the-art methods, even in the case of minimal labeled data. Besides, our pre-trained model exhibits excellent transfer ability for the new downstream traffic classification task.

Acknowledgments

We are grateful to anonymous reviewers for their constructive comments on this work. This work is supported by SJTU-QI'ANXIN Joint Lab of Information System Security.

References

Cai, Z.; Jiang, B.; Lu, Z.; Liu, J.; and Ma, P. 2019. isAnon: Flow-Based Anonymity Network Traffic Identification Using Extreme Gradient Boosting. In *International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Caron, M.; Touvron, H.; Misra, I.; Jégou, H.; Mairal, J.; Bojanowski, P.; and Joulin, A. 2021. Emerging Properties in Self-Supervised Vision Transformers. In *IEEE/CVF International Conference on Computer Visio (ICCV)*, 9630–9640.

Chen, X.; Xie, S.; and He, K. 2021. An Empirical Study of Training Self-Supervised Vision Transformers. In *IEEE/CVF International Conference on Computer Visio (ICCV)*, 9620–9629.

Dadkhah, S.; Mahdikhani, H.; Danso, P. K.; Zohourian, A.; Truong, K. A.; and Ghorbani, A. A. 2022. Towards the Development of a Realistic Multidimensional IoT Profiling Dataset. In *International Conference on Privacy, Security & Trust*.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 4171–4186.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*.

Habibi Lashkari, A.; Draper Gil, G.; Mamun, M.; and Ghorbani, A. 2016. Characterization of encrypted and VPN traffic using time-related features. In *International Conference on Information Systems Security and Privacy*, 407–414.

He, H.; Yang, Z.; and Chen, X. 2020. PERT: Payload Encoding Representation from Transformer for Encrypted Traffic Classification. In *2020 ITU Kaleidoscope: Industry-Driven Digital Transformation*, 1–8.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; and Girshick, R. B. 2022. Masked Autoencoders Are Scalable Vision Learners. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–10.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations (ICLR)*.

Lashkari, A. H.; Draper-Gil, G.; Mamun, M. S. I.; and Ghorbani, A. A. 2017. Characterization of Tor Traffic Using Time Based Features. In *International Conference on Information System Security and Privacy*, 253–262.

Lin, K.; Xu, X.; and Gao, H. 2021. TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of IIoT. *Computer Networks*, 190: 107974.

Lin, X.; Xiong, G.; Gou, G.; Li, Z.; Shi, J.; and Yu, J. 2022. ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification. In *The Web Conference (WWW)*, 633–642.

Liu, C.; He, L.; Xiong, G.; Cao, Z.; and Li, Z. 2019. FS-Net: A Flow Sequence Network For Encrypted Traffic Classification. In *IEEE Conference on Computer Communications (INFOCOM)*, 1171–1179.

Lotfollahi, M.; Siavoshani, M. J.; Zade, R. S. H.; and Saberian, M. 2020. Deep packet: a novel approach for encrypted traffic classification using deep learning. *Soft Comput.*, 24(3): 1999–2012.

Nguyen, T. T.; and Armitage, G. 2008. A Survey of Techniques for Internet Traffic Classification Using Machine Learning. *IEEE Communications Surveys & Tutorials*, 10(4): 56–76.

Panchenko, A.; Lanze, F.; Pennekamp, J.; Engel, T.; Zinnen, A.; Henze, M.; and Wehrle, K. 2016. Website Fingerprinting at Internet Scale. In *Network and Distributed System Security Symposium (NDSS)*, 1–15.

Papadogiannaki, E.; and Ioannidis, S. 2021. A Survey on Encrypted Network Traffic Analysis Applications, Techniques, and Countermeasures. *ACM Computing Surveys*, 54(6).

Shen, M.; Liu, Y.; Zhu, L.; Xu, K.; Du, X.; and Guizani, N. 2020. Optimizing Feature Selection for Efficient Encrypted Traffic Classification: A Systematic Approach. *IEEE Network*, 34(4): 20–27.

Sirinam, P.; Imani, M.; Juárez, M.; and Wright, M. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 1928–1943.

Takase, S.; and Kiyono, S. 2021. Lessons on Parameter Sharing Across Layers in Transformers. *arXiv preprint arXiv:2104.06022*.

Taylor, V. F.; Spolaor, R.; Conti, M.; and Martinovic, I. 2016. AppScanner: Automatic Fingerprinting of Smartphone Apps from Encrypted Network Traffic. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 439–454.

Van Ede, T.; Bortolameotti, R.; Continella, A.; Ren, J.; Dubois, D. J.; Lindorfer, M.; Choffnes, D.; van Steen, M.; and Peter, A. 2020. Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic. In *Network and Distributed System Security Symposium (NDSS)*, volume 27.

Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; and Sheng, Y. 2017. Malware Traffic Classification Using Convolutional Neural Network for Representation Learning. In *International Conference on Information Networking*, 712–717.

Yao, H.; Gao, P.; Zhang, P.; Wang, J.; Jiang, C.; and Lu, L. 2019. Hybrid Intrusion Detection System for Edge-Based IIoT Relying on Machine-Learning-Aided Detection. *IEEE Network*, 33(5): 75–81.

Zhang, J.; Li, F.; Ye, F.; and Wu, H. 2020. Autonomous Unknown-Application Filtering and Labeling for DL-based Traffic Classifier Update. In *IEEE Conference on Computer Communications (INFOCOM)*, 397–405.