ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification

Xinjie Lin[1,2], Gang Xiong[1,2], Gaopeng Gou[1,2], Zhen Li[1,2], Junzheng Shi[1], Jing Yu[1,2*]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of the Chinese Academy of Sciences, Beijing, China

ABSTRACT

Encrypted traffic classification requires discriminative and robust traffic representation captured from content-invisible and imbalanced traffic data for accurate classification, which is challenging but indispensable to achieve network security and network management. The major limitation of existing solutions is that they highly rely on the deep features, which are overly dependent on data size and hard to generalize on unseen data. How to leverage the open-domain unlabeled traffic data to learn representation with strong generalization ability remains a key challenge. In this paper, we propose a new traffic representation model called **E**ncrypted **T**raffic **B**idirectional **E**ncoder **R**epresentations from **T**ransformer (**ET-BERT**), which pre-trains deep contextualized datagram-level representation from large-scale unlabeled data. The pre-trained model can be fine-tuned on a small number of task-specific labeled data and achieves state-of-the-art performance across five encrypted traffic classification tasks, remarkably pushing the F1 of ISCX-VPN-Service to 98.9% (5.2%↑), Cross-Platform (Android) to 92.5% (5.4%↑), CSTNET-TLS 1.3 to 97.4% (10.0%↑). Notably, we provide explanation of the empirically powerful pre-training model by analyzing the randomness of ciphers. It gives us insights in understanding the boundary of classification ability over encrypted traffic. The code is available at: https://github.com/linwhitehat/ET-BERT.

CCS CONCEPTS

• **Information systems** → **Traffic analysis**; • **Security and privacy** → *Network security*; • **Computing methodologies** → *Artificial intelligence.*

KEYWORDS

Encrypted Traffic Classification, Pre-training, Transformer, Masked BURST Model, Same-origin BURST Prediction

ACM Reference Format:
Xinjie Lin[1,2], Gang Xiong[1,2], Gaopeng Gou[1,2], Zhen Li[1,2], Junzheng Shi[1], Jing Yu[1,2*]. 2022. ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3485447.3512217

* Corresponding Author.

WWW '22, April 25–29, 2022, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04.
https://doi.org/10.1145/3485447.3512217

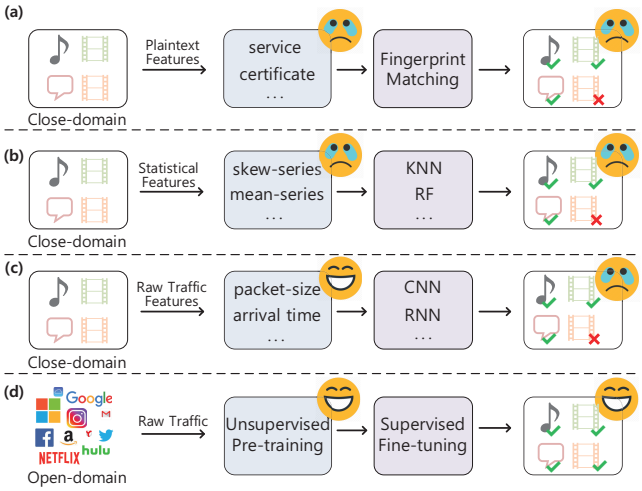


Figure 1: Four main kinds of Encrypted Traffic Classification Methods: (a) Plaintext feature based fingerprint matching. (b) Statistical feature based machine learning. (c) Raw traffic feature based ML. (d) Raw traffic based pre-training.

1 INTRODUCTION

Network traffic classification, aiming to identify the category of traffic from various applications or web services, is an important technique in network management and network security [4, 32]. Recently, traffic encryption has been widely utilized to protect the privacy and anonimity of Internet users. However, it also brings great challenges to traffic classification since the malware traffic and the cybercriminals can evade the surveillance system by privacy-enhanced encryption techniques, such as Tor, VPN, etc. Traditional methods capture patterns and keywords in the data packets from the payload, called deep packet inspection (DPI), fail to apply to the encrypted traffic. Furthermore, due to the rapid development of encryption technology, traffic classification methods for a specific kind of encrypted traffic cannot adapt well to the new environment or unseen encryption strategies [27]. Therefore, how to capture the implicit and robust patterns in the diverse encrypted traffic and support accurate and generic traffic classification is essential to achieve high network security and effective network management.

To tackle the above problem, research in encrypted traffic classification has evolved significantly over time as illustrated in Figure 1. Early works [35] leverage the remaining plaintext in the encrypted traffic (*e.g.* certificates) to construct the fingerprint and conduct fingerprint matching for classification (Figure 1(a)). However, these methods are not applicable to the newly emerging encrypted techniques (*e.g.* TLS 1.3) since the plaintext becomes more sparse or

obfuscated. To this end, some works [25, 34] extract the statistical feature and employ classical machine learning algorithms to handle the encrypted traffic without plaintext (Figure 1(b)). These methods highly rely on expert-designed features and have limited generalization ability. Recently, deep learning methods [19, 20] automatically learn complicated patterns from the raw traffic (Figure 1(c)), and achieve remarkable performance improvement. However, these methods highly rely on the amount and distribution of labelled training data, which is easy to cause model bias and hard to adapt to newly emerged encryption.

In recent years, pre-training models have great breakthrough in nature language processing [6], computer vision [8] and a wide range of other fields [2, 17]. Pre-training based methods adopt large unlabeled data to learn the unbiased data representations. Such data representations can be easily transferred to the downstream tasks by fine-tuning on limited amount of labeled data. In the field of encrypted traffic classification, the most recent work [12] directly applies the pre-training technique and obtains obvious improvement on VPN traffic classification, but it lacks a pre-training task designed for traffic and a reasonable input representation to demonstrate the effect of the pre-training model.

In this paper, we propose a novel pre-training model for classifying encrypted traffic, called **E**ncrypted **T**raffic **B**idirectional **E**ncoder **R**epresentations from **T**ransformer (ET-BERT). It aims to learn generic traffic representations from large-scale unlabeled encrypted traffic (Figure 1(d)). We first propose a raw traffic representation model to transform the datagram to language-like tokens for pre-training. Each traffic flow is presented by a transmission-guided structure, denoted as BURST, which serves as the input. The proposed framework consists of two stages: pre-training and fine-tuning. Specifically, the pre-training network with Transformer structure obtains datagram-level generic traffic representations by self-supervised learning on large-scale unlabeled encrypted traffic. Thereinto, we propose two novel pre-training tasks to learn the traffic-specific patterns: the Masked BURST Model (MBM) task captures the correlated relationship between different datagram bytes in the same BURST and represent them by their context; the Same-origin BURST Prediction (SBP) task models the transmission relationships of preceding and subsequent BURST. Then, ET-BERT incorporates with the specific classification task and fine-tune the parameters with small number of task-specific labeled data.

The main contributions of this paper are summarized as follows: (1) We propose a pre-training framework for encrypted traffic classification, which leverages large-scale unlabeled encrypted traffic to learn generic datagram representation for a series of encrypted traffic classification tasks. (2) We newly propose two traffic-specific self-supervised pre-training tasks, *e.g.* Masked BURST Model and Same-origin BURST Prediction, which capture both byte-level and BURST-level contextual relationships to obtain generic datagram representations. (3) ET-BERT has great generalization ability and achieves a new state-of-the-art performance over 5 encrypted traffic classification tasks, including General Encrypted Application Classification, Encrypted Malware Classification, Encrypted Traffic Classification on VPN, Encrypted Application Classification on Tor, Encrypted Application Classification on TLS 1.3, and outperforms existing works remarkably by 5.4%, 0.2%, 5.2%, 4.4%, 10.0%. Meanwhile, we provide theoretical explanation and analysis on the powerful performance of the pre-trained model.

2 RELATED WORK

2.1 Encrypted Traffic Classification

Fingerprint Construction. Unlike the packet-inspection approach under plain-text traffic, which fails when the traffic is encrypted, some studies suggest using unencrypted protocol field information. FlowPrint [35] extracts device, certificate, size, and temporal features to represent each flow and constructs a fingerprint library by clustering and cross-correlating for efficient traffic classification. However, these fingerprints are easily tampered with in virtual communication networks and lose their correct meaning, whereas our model does not rely on any plain-text information.

Statistical Methods. Most studies of encrypted traffic exploit the statistical properties of the traffic to be independent of traffic encryption. AppScanner [34] exploits statistical features of packet size for training random forest classifiers, while BIND [1] also exploits statistical features of temporality. However, it is hard to design generic statistical features to cope with the massive applications and websites that keep becoming complex, while our model does not need to rely on human-designed features.

Deep Learning Models. Encrypted traffic classification using supervised deep learning have become a popular approach that automatically extracts discriminative features rather than relying on manual design. DF [33] uses convolutional neural networks (CNNs) and FS-Net [20] uses recurrent neural networks (RNNs) to automatically extract representations from raw packet size sequences of encrypted traffic, while Deeppacket [23] and TSCRNN [19] are characterizing raw payloads. However, this approach relies on a large amount of supervised data to capture valid features thus learning biased representations in imbalanced data, while our model does not rely on large labeled data.

2.2 Pre-training Models

In natural language processing, the deep bidirectional pre-training model based on Transformers achieves the best results for multiple tasks. With this representation type and structure, RoBERTa [22] uses dynamic masking and ALBERT [16] proposes sentence order prediction to improve performance by advancing unsupervised tasks. The extensions of the pre-training models include knowledge enhancement and model compression, ERNIE [40] introduces entity knowledge to improve language understanding, while DistilBERT [28] reduces the number of network layers and parameters through knowledge distillation techniques to significantly speed up model training but with a slight reduction in performance. In addition, the wide applications of pre-training models in cross-domains such as visual language as well as computer vision demonstrate their advantages of utilizing unlabeled data to help learn robust feature representations on limited labeled data.

In encrypted traffic classification, although payloads have no semantics, Sengupta *et al.* [29] exploit the randomness difference between different ciphertexts to distinguish different applications, which suggests that the encrypted traffic is not perfectly random and implicit patterns exist. PERT [12] first applies the pre-training model to migrate ALBERT to encrypted traffic classification and

achieves 93.23% performance in ISCX-VPN-Service [9] on F1. However, it lacks of specific design for encrypted traffic representation and the corresponding pre-training tasks, which limits its generalization ability on new encryption techniques (*e.g.* TLS 1.3) according to our empirical study in Section 4.2. We design two pre-training tasks taking into account the structural pattern of traffic transmission and the bi-directional association of packet payloads, then use two fine-tuning strategies to better fit the traffic classification tasks.

3 ET-BERT

3.1 Model Architecture

In this paper, we aim to learn generic encrypted traffic representations and classify them in different scenarios (*e.g.* applications, encryption protocols, or services). To this end, our proposed pre-training strategy contains two main stages: pre-training for learning generic encrypted traffic representations with large-scale unlabelled data and fine-tuning for adjusting the pre-trained model for the specific downstream task. In the pre-training stage, given the unlabelled traffic flows, the pre-trained model outputs datagram-level generic traffic representations. In the fine-tuning stage, given the target-specific labelled packets or flows, the fine-tuned model predicts its category.

Encrypted traffic differs greatly from nature language and images in that it doesn't contain human-understandable content and explicit semantic units. In order to effectively leverage the pre-training technique for encrypted traffic classification, we mainly propose three main components in ET-BERT as shown in Figure 2: (1) We propose *Datagram2Token* approach (Section 3.2) to transform encrypted traffic to pattern-preserved token unit for pre-training; (2) Then two pre-training tasks, *e.g.* Masked BURST Model and Same-origin BURST Prediction, are proposed to learn the contextualized datagram representations from the transition context instead of the semantic context (Section 3.3); (3) To adapt to different traffic classification scenarios, we further propose two fine-tuning strategies, *e.g.* packet-level fine-tuning for single packet classification and flow-level fine-tuning for single flow classification (Section 3.4).

The main network architecture of ET-BERT consists of multi-layer bi-directional Transformer blocks [36]. Each of the block is composed of multi-head self-attention layers, which captures the implicit relationships between the encoded traffic units in datagrams. In this work, the network architecture consists of 12 transformer blocks with 12 attention heads in each self-attention layer. The dimension of each input token H is set to 768 and the number of input tokens is 512.

3.2 Datagram2Token Traffic Representation

In the real network environment, huge amount of traffic contains diverse flows of different categories (*e.g.* different applications, protocols or services), which makes it difficult to learn a stable and discriminative representation of a certain kind of traffic. Therefore, we first split out flows with the same IP, port and protocol from the traces before representing traffic. As a result, each splitted flow comes from the same traffic category containing a complete flow session. To further transform a flow into a word-like tokens similar to nature language, we propose a *Datagram2Token* module that consists of three processes: (1) *BURST Generator* extracts continues server-to-client or client-to-server packets in one session flow, named as BURST [26, 31], to represent the partial complete information of a session. (2) Then *BURST2Token* process transforms the datagram in each BURST to token embeddings via the bi-gram model. Meanwhile, this process also splits a BURST into two segments preparing for the pre-training tasks. (3) Finally, *Token2Emebdding* concatenates the token embedding, position embedding and segmentation embedding of each token to serve as the input representation for pre-training.

3.2.1 BURST Generator. A BURST is defined as a set of time-adjacent network packets originated from either the request or the response in a single session flow. A sequence of BURSTs characterize the pattern of network flow transmission from the application layer perspective. In the application layer, the Document Object Model (DOM) tree between web pages becomes diverse, stemming from the personalization of web services. As the client-side rendering process divides the web data into different objects (*e.g.* text and images), the DOM structure generates semantic-aware fragments and subliminally affects the client's resource requests. Each generated segment forms a BURST of network, which contains a complete part of content with specific type from the DOM structure. We extract the BURSTs as input for the pre-training model.

For the BURST, we are concerned with the source and destination of the packet. Given a trace from packet capture file as a sequence $Trace = \{flow_i, i \in \mathbb{N}^+\}$, where $flow = \{p_j, j \in \mathbb{N}^+\}$ is a session flow consisting of source-to-destination packets p identified by a five-tuple (IPsrc:PORTsrc, IPdst:PORTdst, Protocol). The BURST is defined as:

$$BURST = \begin{cases} B^{src} = \{p_m^{src}, m \in \mathbb{N}^+\} \\ B^{dst} = \{p_n^{dst}, n \in \mathbb{N}^+\} \end{cases} \quad (1)$$

where m, n denotes the maximum number of unidirectional packets of source-to-destination and destination-to-source respectively.

3.2.2 BURST2Token. In order to transform the BURST representation into the token representation for pre-training, we decompose the hexadecimal BURST into a sequence of units.

To this end, we use a bi-gram to encode the hexadecimal sequence, where each unit consists of two adjacent bytes. We then use Byte-Pair Encoding for token representation, where each token unit ranges from 0 to 65535, the dictionary size $|V|$ is max expressed as 65536. In addition, we also add the special tokens [CLS], [SEP], [PAD] and [MASK] for training tasks. The first token of each sequence is always [CLS], and the final hidden layer state associated with this token is used to represent the complete sequence for classification tasks. The token [PAD] is a padding notation to satisfy the minimum length requirement. The sub-BURST pair of a BURST will be separated by [SEP]. The token [MASK] appears during pre-training to learn the context of the traffic.

As shown in Figure 2, we equally divided a BURST into two sub-BURSTs for SBP task. We differentiate the sub-BURSTs by the special token [SEP] and the segment embedding indicating whether it belongs to segment A or segment B. We denote the segment A as sub-BURST^A and the segment B as sub-BURST^B.

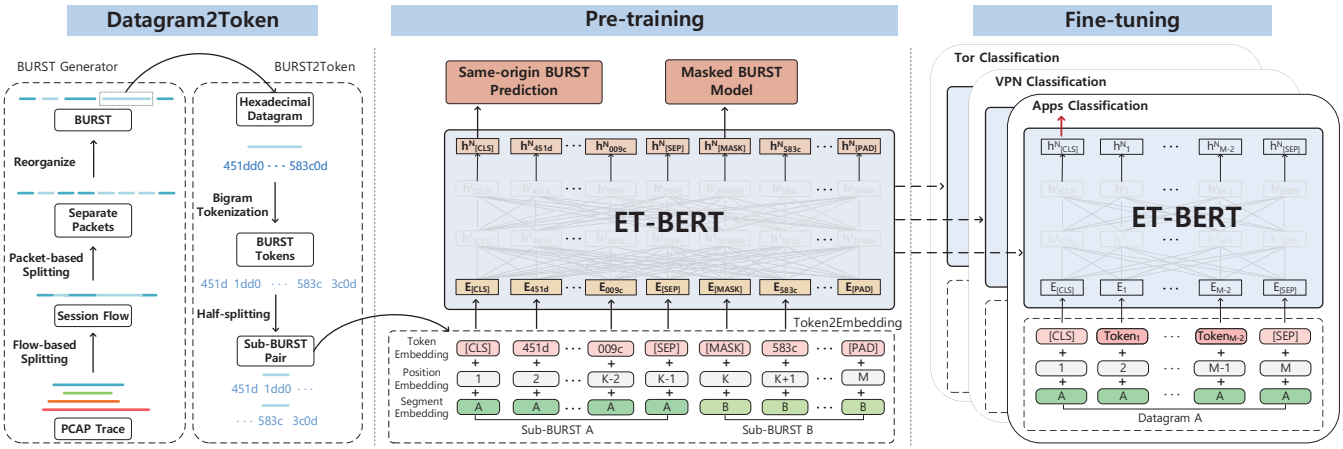


Figure 2: Overview of ET-BERT Framework.

3.2.3 Token2Embedding. We represent each token obtained in BURST2Token by three embeddings: token embedding, position embedding and segment embedding. A full token representation is constructed by summing up the aforementioned three embeddings. In this work, we take the full tokenized datagrams as original inputs. The first group of embedding vectors are randomly initialised, where the embedding dimension is $D = 768$. After N times of Transformer encoding, we obtain the final token embedding.

Token Embedding. As shown in Figure 2, the representation of the token learn from the lookup table in Section 3.2.2 is called token embedding *Etoken*. The final hidden vector of the input token as $Etoken \in \mathbb{R}^H$, where the embedding dimension H is set to 768.

Position Embedding. Since the transmission of traffic data is strongly related to the order, we use position embedding to ensure the model learn to focus on the temporal relationship of tokens by relative positions. We assign an H-dimensional vector to each input token for representing its position information in the sequence. We denote the position embedding as $Epos \in \mathbb{R}^H$, where the embedding dimension H is set to 768.

Segment Embedding. As mentioned in Section 3.2.2, the segment embedding of sub-BURST is denoted as $Eseg \in \mathbb{R}^H$, where the embedding dimension H is set to 768. At the fine-tuning stage, we represent a packet or a flow as one segment for classification task.

3.3 Pre-training ET-BERT

Our proposed two pre-training tasks capture the contextual relationship between traffic bytes by predicting the masked token as well as the correct transmission order by predicting the Same-origin BURST. The detailed process is shown in the middle of Figure 2.

Masked BURST Model. This task is similar to the Masked Language Model utilized by BERT [6]. The key difference is that traffic tokens without obvious semantics are incorporated into ET-BERT for capturing the dependencies among datagram bytes. During the pre-training, each token in the input sequence is randomly masked with 15% probability. As the chosen token, we replace it with [MASK] at 80% chance, or choose a random token to replace it or leave it unchanged at 10% chance, respectively.

For the masked tokens are replaced by the special token [MASK], ET-BERT is trained to predict tokens at the masked positions based on the context. Benefiting from the deep bi-directional representation brought by this task, we randomly mask k tokens for the input sequence X. We use the negative log likelihood as our loss function and formally define it as:

$$L_{MBM} = -\sum_{i=1}^{k} \log(P(MASK_i = token_i|\bar{X};\theta)) \quad (2)$$

where θ represents the set of trainable parameters of ET-BERT. The probability P is modeled by the Transformer encoder with θ. $\bar{X}$ is the representation of X after masking and $MASK_i$ represents the masked token at the i_th position in the token sequence.

Same-origin BURST Prediction. The importance of BURSTs in network traffic has been declared in the previous section, and our purpose is to better learn the traffic representations by capturing the correlation of packets in BURSTs. Moreover, we consider the tight relationship between BURST structure and the web content, which is able to convey the difference between BURSTs generated from different categories of traffic. For example, there is a differentiation in the traffic by loading content separately for social networking sites with different DOM structures, *e.g.* in the order of text, image, video and in the order of image, text, video. This phenomenon was also confirmed by the study [37] for intra-domain fingerprinting.

We learn the dependencies between packets inside BURST via the Same-origin BURST Prediction (SBP) task. For this task, a binary classifier is used to predict whether two sub-BURST are from the same BURST origin. Specifically, when choosing the sub-BURST^A and sub-BURST^B for each sub-BURST pair, 50% of the time sub-BURST^B is the actual next sub-BURST that follows sub-BURST^A, and 50% of the time it is a random sub-BURST from other BURSTs. For a given input containing sub-BURST pair $B_j = (sub\text{-}B_j^A, sub\text{-}B_j^B)$ and its ground-truth label $y_j \in [0, 1]$ (0 represents paired sub-BURSTs and 1 represents unpaired ones).

$$L_{SBP} = -\sum_{j=1}^{n} \log(P(y_j|B_j;\theta)) \quad (3)$$

Overall, the final pre-training objective is the sum of the above two losses, which is defined as:

$$L = L_{MBM} + L_{SBP} \tag{4}$$

Pre-training Dataset. In this work, around 30GB of unlabeled traffic data is used for pre-training. This dataset contains two parts: (1) about 15GB traffic from the public datasets [9, 30]; (2) about 15GB traffic from our passively collected traffic under the China Science and Technology Network (CSTNET). Further, the dataset contains rich network protocols, such as a new encryption protocol based on UDP transport QUIC, Transport Layer Security, File Transfer Protocol, Hyper Text Transfer Protocol, Secure Shell, etc., which are common network protocols.

3.4 Fine-tuning ET-BERT

Fine-tuning can serve downstream classification tasks well because: (1) the pre-training representation is traffic class-independent and can be applied to any class of traffic representation; (2) since the input of the pre-training model is at the datagram bytes level, downstream tasks that need to classify packets and flows can be transformed into the corresponding datagram byte token to be classified by the model; (3) the special [CLS] token of the output of the pre-training model models the representation of the entire input traffic and can be employed directly for classification.

Since the structure of fine-tuning and pre-training is basically identical, we input the task-specific packet or flow representations into the pre-trained ET-BERT and fine-tune all parameters in an end-to-end model. At the output layer, the [CLS] represenation is fed to a multi-class classifier for prediction. We propose two fine-tuning strategies to adapt the classification of different scenarios: (1) packet level as input dedicated to experimenting whether ET-BERT can adapt to more fine-grained traffic data, as **ET-BERT(packet)**; (2) flow level as input dedicated to fairly and objectively comparing ET-BERT with other methods, as **ET-BERT(flow)**. The major difference between the two fine-tuning models is the amount of information of the input traffic. We use a stitched datagram of M consecutive packets in a flow as input data, where M is set to 5 in our approach. The traffic data processing is described in detail in Section 4.1.

The cost of fine-tuning is relatively cheap compared to pre-training, and a single GPU is sufficient for a fine-tuning task.

4 EXPERIMENTS

In this section, we conduct five encrypted traffic classification tasks (Section 4.1) to prove the effectiveness of ET-BERT to solve problems of different encryption scenarios and imbalanced data distribution. We then compare our model with 11 methods (Section 4.2) and perform an ablation analysis of the key components of the model (Section 4.3). We further provide an interpretative analysis of the remarkable performance obtained by ET-BERT (Section 4.4), and the ability to handle few-shot samples (Section 4.5).

4.1 Experiment Setup

4.1.1 Datasets and Downstream Tasks. To evaluate the effectiveness and generalization of ET-BERT, we conduct experiments across five encrypted traffic classification tasks on six public datasets and one newly proposed dataset. The tasks and the corresponding datasets are shown in Table 1.

Table 1: The Statistical Information of the Datasets.

Task	Dataset	#Flow	#Packet	#Label
GEAC	Cross-Platform(iOS) [35]	20,858	707,717	196
	Cross-Platform(Android) [35]	27,846	656,044	215
EMC	USTC-TFC [39]	9,853	97,115	20
ETCV	ISCX-VPN-Service [9]	3,694	60,000	12
	ISCX-VPN-App [9]	2,329	77,163	17
EACT	ISCX-Tor [10]	3,021	80,000	16
EAC-1.3	CSTNET-TLS 1.3 (Ours)	46,372	581,709	120

Task 1: General **E**ncrypted **A**pplication **C**lassification (**GEAC**) task aims to classify application traffic under standard encryption protocols. We test on Cross-Platform (iOS) [35] and Cross-Platform (Android) [35], which contain 196 and 215 applications respectively. The iOS apps and the Android apps were collected from the top 100 Apps from the US, China and India. This dataset with the largest number of categories and long-tail data distribution over all classes.

Task 2: Encrypted **M**alware **C**lassification (**EMC**) is a collection of encrypted traffic consisting of malware and benign applications [39]. The dataset USTC-TFC [39] contains 10 categories of benign traffic and 10 categories of malicious traffic.

Task 3: Encrypted **T**raffic **C**lassification on **VPN** (**ETCV**) task classifies encrypted traffic that uses Virtual Private Networks (VPNs) for network communication. VPNs are popular for bypassing censorship as well as accessing geo-locked services, which is difficult to detect due to its protocol obfuscation. We use the commonly compared ISCX-VPN [9], which is constructed of 6 communication applications captured through the Canadian Institute for Cybersecurity in both VPN and non-VPN. To test ET-BERT on service and application, we further categorize the dataset by services and applications, forming the ISCX-VPN-Service dataset with 12 categories and the ISCX-VPN-App dataset with 17 applications.

Task 4: Encrypted **A**pplication **C**lassification on **Tor** (**EACT**) task aims to classify encrypted traffic that uses the Onion Router (Tor) for communication privacy enhancement. The dataset [10] is called ISCX-Tor, which contains 16 applications. This kind of traffic further obscures the behavior of the traffic by obfuscating the communication between the sender and the receiver through a distributed routing network, which is more challenging for traffic classification as the pattern extraction of traffic becomes harder.

Task 5: Encrypted **A**pplication **C**lassification on **TLS 1.3** (**EAC-1.3**) task aims to classify encrypted traffic over new encryption protocol TLS 1.3. The dataset is our collection of 120 applications under CSTNET from March to July 2021, named as CSTNET-TLS 1.3. As we know, this is the first TLS 1.3 dataset to date. The applications are acquired from Alexa Top-5000 [3] deployed with TLS 1.3 and we label each session flow by the server name indication (SNI). In CSTNET-TLS 1.3, the SNI remains accessible due to the compatibility of TLS 1.3. The ECH mechanism will disable the SNI in the future and compromise the accuracy of the labeling, but we discuss some thoughts to overcome it in Section 5.

Table 2: Comparison Results on Cross-Platform, ISCX-VPN-Service and ISCX-VPN-App datasets.

Dataset	Cross-Platform(iOS)				Cross-Platform(Android)				ISCX-VPN-Service				ISCX-VPN-App			
Method	AC	PR	RC	F1	AC	PR	RC	F1	AC	PR	RC	F1	AC	PR	RC	F1
AppScanner[34]	0.3205	0.2103	0.2173	0.2030	0.3868	0.2523	0.2594	0.2440	0.7182	0.7339	0.7225	0.7197	0.6266	0.4864	0.5198	0.4935
CUMUL[25]	0.2910	0.1917	0.2081	0.1875	0.3525	0.2221	0.2409	0.2189	0.5610	0.5883	0.5676	0.5668	0.5365	0.4129	0.4535	0.4236
BIND[1]	0.3770	0.2566	0.2715	0.2484	0.4728	0.3126	0.3253	0.3026	0.7534	0.7583	0.7488	0.7420	0.6767	0.5152	0.5153	0.4965
K-fp[11]	0.2155	0.2037	0.2069	0.2003	0.2248	0.2113	0.2104	0.2052	0.6430	0.6492	0.6417	0.6395	0.6070	0.5478	0.5430	0.5303
FlowPrint[35]	0.9254	0.9438	0.9254	0.9260	0.8698	0.9007	0.8698	0.8702	0.7962	0.8042	0.7812	0.7820	0.8767	0.6697	0.6651	0.6531
DF[33]	0.3106	0.2232	0.2179	0.2140	0.3862	0.2595	0.2620	0.2527	0.7154	0.7192	0.7104	0.7102	0.6116	0.5706	0.4752	0.4799
FS-Net[20]	0.3712	0.2845	0.2754	0.2655	0.4846	0.3544	0.3365	0.3343	0.7205	0.7502	0.7238	0.7131	0.6647	0.4819	0.4848	0.4737
GraphDApp[31]	0.3245	0.2450	0.2392	0.2297	0.4031	0.2842	0.2786	0.2703	0.5977	0.6045	0.6220	0.6036	0.6328	0.5900	0.5472	0.5558
TSCRNN[19]	-	-	-	-	-	-	-	-	-	0.9270	0.9260	0.9260	-	-	-	-
Deeppacket[23]	0.9204	0.8963	0.8872	0.9034	0.8805	0.8004	0.7567	0.8138	0.9329	0.9377	0.9306	0.9321	0.9758	0.9785	0.9745	0.9765
PERT[12]	0.9789	0.9621	0.9611	0.9584	0.9772	0.8628	0.8591	0.8550	0.9352	0.9400	0.9349	0.9368	0.8229	0.7092	0.7173	0.6992
ET-BERT(flow)	**0.9844**	0.9701	0.9632	0.9643	**0.9865**	0.9324	**0.9266**	**0.9246**	0.9729	0.9756	0.9731	0.9733	0.8519	0.7508	0.7294	0.7306
ET-BERT(packet)	0.9810	**0.9757**	**0.9772**	**0.9754**	0.9728	**0.9439**	0.9119	0.9206	**0.9890**	**0.9891**	**0.9890**	**0.9890**	**0.9962**	**0.9936**	**0.9938**	**0.9937**

Table 3: Comparison Results on ISCX-Tor, USTC-TFC and CSTNET-TLS 1.3 datasets.

Dataset	ISCX-Tor				USTC-TFC				CSTNET-TLS 1.3			
Method	AC	PR	RC	F1	AC	PR	RC	F1	AC	PR	RC	F1
AppScanner[34]	0.6722	0.3756	0.4422	0.3913	0.8954	0.8984	0.8968	0.8892	0.6662	0.6246	0.6327	0.6201
CUMUL[25]	0.6606	0.3850	0.4416	0.3918	0.5675	0.6171	0.5738	0.5513	0.5391	0.4942	0.5060	0.4904
BIND[1]	0.7185	0.4598	0.4515	0.4511	0.8457	0.8681	0.8382	0.8396	0.7964	0.7605	0.7650	0.7560
K-fp[11]	0.6472	0.5576	0.5849	0.5522	-	-	-	-	0.4036	0.3969	0.4044	0.3902
FlowPrint[35]	0.9092	0.3820	0.3661	0.3654	0.8146	0.6434	0.7002	0.6573	0.1261	0.1354	0.1272	0.1116
DF[33]	0.7533	0.6228	0.6010	0.5850	0.7787	0.7883	0.7819	0.7593	0.7936	0.7721	0.7573	0.7602
FS-Net[20]	0.6071	0.5080	0.5350	0.4590	0.8846	0.8846	0.8920	0.8840	0.8639	0.8404	0.8349	0.8322
GraphDApp[31]	0.6836	0.4864	0.4823	0.4488	0.8789	0.8226	0.8260	0.8234	0.7034	0.6464	0.6510	0.6440
TSCRNN[19]	-	0.9490	0.9480	0.9480	-	0.9870	0.9860	0.9870	-	-	-	-
Deeppacket[23]	0.7449	0.7549	0.7399	0.7473	0.9640	0.9650	0.9631	0.9641	0.8019	0.4315	0.2689	0.4022
PERT[12]	0.7682	0.4424	0.4446	0.4345	0.9909	0.9911	0.9910	0.9911	0.8915	0.8846	0.8719	0.8741
ET-BERT(flow)	0.8311	0.5564	0.6448	0.5886	**0.9929**	**0.9930**	**0.9930**	**0.9930**	0.9510	0.9460	0.9419	0.9426
ET-BERT(packet)	**0.9921**	**0.9923**	**0.9921**	**0.9921**	0.9915	0.9915	0.9916	0.9916	**0.9737**	**0.9742**	**0.9742**	**0.9741**

Ethical Considerations. For this research, an IRB was consulted and any identification was not utilized. Furthermore, the collection was completely passive. We have conformed to the user agreements of the corporate network where the data was collected.

4.1.2 Data Pre-processing. We remove packets of Address Resolution Protocol (ARP) and Dynamic Host Configuration Protocol (DHCP), which are irrelevant to specific traffic of the transmitted content. To avoid the influence of the packet header, which may introduce biased interference in a finite set with strong identification information such as IP and port [18, 23, 38], we removed the Ethernet header, the IP header, and protocol ports of the TCP header. In the stage of fine-tuning, we randomly select at most 500 flows and 5,000 packets from each class in all datasets. Each dataset is divided into the training set, the validation set and the testing set according to the ratio of 8 : 1 : 1.

4.1.3 Evaluation Metrics and Implementation Details. We evaluate and compare the performance of the our model by four typical metrics, including Accuracy (AC), Precision (PR), Recall (RC), and F1 [35, 42]. Macro Average [21] is used to avoid biased results due to imbalance between multiple categories of data by calculating the mean value of AC, PR, RC and F1 of each category. In pre-training, the batch size is 32 and the total steps is 500,000. We set the learning rate is 2×10^{-5}, and the ratio of warmup is 0.1. We fine-tune with the AdamW optimizer for 10 epochs, where the learning rate is set to 6×10^{-5} for flow-level, and 2×10^{-5} for packet-level. The batch size is 32 and the dropout rate is 0.5. All the experiments are implemented with Pytorch 1.8.0 and UER [41], conducted with NVIDIA Tesla V100S GPUs.

4.2 Comparison with State-of-the-Art Methods

We compare ET-BERT with various state-of-the-art (SOTA) methods, including (1) fingerprint construction method: FlowPrint [35]; (2) statistical feature methods: AppScanner [34], CUMUL [25], BIND [1] and k-fingerprinting (K-fp) [11]; (3) deep learning methods: Deep Fingerprinting (DF) [33], FS-Net [20], GraphDApp [31], TSCRNN [19], Deeppacket [23]; (4) pre-training method: PERT [12]. The experimental results are shown in Tables 2 and 3. Additional comparison study can be found in Appendix A.1.

GEAC. According to Table 2, both ET-BERT(packet) and ET-BERT(flow) outperform all methods significantly. Our model obtains 1.7% and 5.4% respective improvement from the existing state of the art ((e.g.) PERT and FlowPrint) on Cross-Platform (iOS) and Cross-Platform (Android). FlowPrint utilizes plain-text fingerprints including certificate fields to build a multi-dimensional fingerprint library for identification of applications. However, ET-BERT learns

Table 4: Ablation Study of Key Components in ET-BERT on ISCX-VPN-App dataset.

	Method	SBP	MBM	PT-P	PT-B	FT-f	FT-cf	FT-P	AC	PR	RC	F1
ET-BERT(packet)(full model)		✓	✓	×	✓	×	×	✓	**0.9471**	**0.9462**	**0.9412**	**0.9395**
1	w/o SBP	×	✓	×	✓	×	×	✓	0.9000	0.9142	0.9000	0.8998
2	w/o MBM	✓	×	×	✓	×	×	✓	0.8471	0.8666	0.8471	0.8462
3	w/o BURST	✓	✓	✓	×	×	×	✓	0.9235	0.9386	0.9235	0.9258
4	ET-BERT(flow)	✓	✓	×	✓	✓	×	×	0.8133	0.7661	0.7374	0.7387
5	concatenated-flow(cf)	✓	✓	×	✓	×	✓	×	0.8229	0.7488	0.6812	0.6961
6	w/o pre-training(packet)	×	×	×	×	×	×	✓	0.5882	0.6152	0.5882	0.5638

contextual relationships on ciphertext without relying on any plaintext fields. In addition, our model learns the pattern of traffic transmission structure while PERT has not been able to master.

EMC. From the results on USTC-TFC as presented in Table 3, we can observe that the performance of all the methods are inferior to our model.Our model achieves the best performance on F1 as 99.30%. We observe that the malicious traffic in this dataset contain unencrypted data in application layer and this makes much easier for other models to leverage such plaintext for easier classification.

ETCV ET-BERT achieves 5.69% and 1.72% improvement over the existing state-of-the-art model Deeppacket on ISCX-VPN-Service and ISCX-VPN-App. Both datasets raise the imbalance challenge, which is more severe on ISCX-VPN-App. Our model and Deeppacket alleviate the effect of imbalanced data by learning correlations between packet datagrams. Besides, ET-BERT achieves an average improvement on F1 of 25.55% and 42.89% for all methods except PERT, which indicates that our model has strong ability in identifying confusion traffic even in the case of imbalanced data.

EACT. As the results on ISCX-Tor shown in Table 3, ET-BERT improves 4.41% over the existing best result obtained by TSCRNN. The initial traffic in Tor is not only multi-layer encrypted but also adversarially obfuscated. TSCRNN enriches flows by random sampling to train the model, while we exploit the intrinsic relationship of packets to achieve better classification.

EAC-1.3. Our model is improved by 10.0% from 87.41% over the existing state-of-the-art as the results on CSTNET-TLS 1.3 shown in Table 3. TLS 1.3 poses new challenge for the FlowPrint and Deeppacket by improving the security of the transmission and conversely. ET-BERT pushes F1 to 97.41% by deeply representing datagrams. This indicates that the encrypted traffic datagrams over TLS 1.3 still have implicit patterns, which is better leveraged by ET-BERT for classification.

4.3 Ablation Study

We show ablation results to verify the contribution of each component on the widely compared ISCX-VPN-App. To fairly compare packet and flow level fine-tuning approaches, we randomly selected at most 100 packets and flows from each class as the training dataset. In Table 4, PT-P and PT-B respectively represent the inputs are randomly selected adjacent packets and our proposed BURST packets in pre-training (PT). FT-f, FT-cf and FT-P respectively denote using a flow, a concatenated flow [12] and a single packet in fine-tuning (FT). (1) In models '1-3', we evaluate the impact of each task and the input of pre-training. The respective decrease of 3.97% and 9.33% on F1 for '1' and '2' indicates that both self-supervised tasks are beneficial in providing complementary patterns for classification. In addition, we input packets instead of BURST in '3' and the F1 score decreases by 1.37%. It proves that the BURST structure can learn the relationship between packets for better traffic classification. (2) In model '4' and '5', we evaluate the effect of fine-tuning flows in different forms. Model '4' uses consecutive packets as the input while model '5' uses packets separately as the input and concatenates the outputs at the final encoder layer, as like PERT [12]. When we switch from flow to concatenated-flow, the results for model '5' drop by 4.26%. Different packets of one flow are interdependent and our fine-tuning method for classifying flows is more beneficial. (3) We remove the pre-trained model to evaluate the impact of pre-training. According to model '6', we perform supervised learning on labeled data by training the Transformer model directly and the F1 score decreases remarkably by 37.57% compared with ET-BERT.

Table 5: Results of 15 Randomness Tests on 5 Ciphers.

Ciphers/Tests	AES(GCM)	AES(CBC)	CHA20	ARC4	3DES
Monobit	0.7918	0.2585	0.9761	0.5687	0.4099
Block Frequency	0.6316	0.0791	0.0176	0.4821	0.6434
Independent Runs	0.8824	0.1672	0.8966	0.7052	0.4241
Longest Runs	0.7198	0.3148	0.5134	0.5156	0.2889
Spectral	0.6202	0.9707	0.9415	0.6729	0.5756
Overlapping Patterns(OP)	0.0519	0.9856	0.1002	0.9089	0.4762
Non OP	0.8148	0.1967	0.4445	0.0096	0.5156
Universal	0.8501	0.3277	0.1149	0.0416	0.3062
Serial	0.7690	0.4539	0.1600	0.6068	0.8381
Approximate Entropy	0.9239	0.5226	0.3371	0.3470	0.3611
Cumulative Sums	0.9496	0.4512	0.7355	0.1742	0.4043
Random Excursions(RE)	0.1811	0.1232	0.4112	0.9424	0.9091
RE Variant	0.4805	0.0119	0.9542	0.5978	0.9065
Matrix Rank	0.5674	0.4890	0.0880	0.0504	0.1447
Linear Complexity	0.6235	0.4519	0.7428	0.0952	0.9384

4.4 Interpretability

4.4.1 Randomness Analysis. The aforementioned results demonstrate the effectiveness and generalization of ET-BERT due to the imperfect randomness of the ciphers of encrypted payloads. An ideal encryption scheme causes the generated message to bear the maximum possible entropy [5]. However, this hypothesis is not

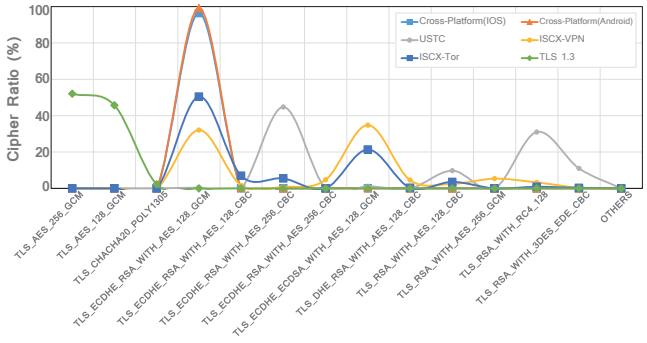


Figure 3: The Distribution of the Ciphers across all Datasets.

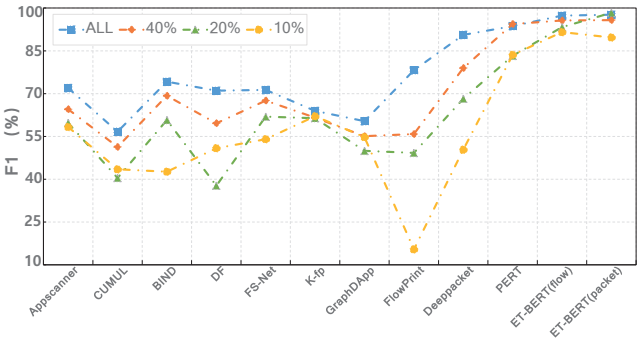


Figure 4: Comparison Results on Few-shot ISCX-VPN-App.

valid in practice since different cipher implementations have varying degrees of randomness [7]. We evaluate the strength of 5 ciphers in this paper through 15 sets of statistical tests[24], where *p-value* = 1 indicates perfect randomness of the sequences. As shown in Table 5, these ciphers indeed fail to achieve perfect randomness.

4.4.2 Impact of Ciphers. To assess the impact of the difference ciphers of ET-BERT, we analyse the employment of ciphers for different datasets. As shown in Figure 3, the horizontal coordinate indicates the ciphers that account for the top 13 types and others, and the vertical coordinate represents the percentage of each cipher. The ISCX-VPN, ISCX-Tor and USTC-TFC contain at least 3 ciphers including RC4 and 3DES with weaker randomness, while other datasets mainly consist of one cipher. According to Tables 2 and 3, ET-BERT achieves an F1 close to 100% on datasets with weaker randomness and the presence of greater fluctuations, average 99.14% in ETCV, 99.21% in EACT, and 99.30% in EMC.

4.5 Few-shot Analysis

To validate the effectiveness and robustness of ET-BERT in few-shot settings, we design comparison experiments with different data proportions on ISCX-VPN-Service. We set the data size of each category to 500 and randomly select 40%, 20% and 10% of the samples for the few-shot experiments. In Figure 4, the comparison results illustrate that the pre-training method is least affected by the reduction of data size. The F1 scores of ET-BERT(packet) with 40%, 20%, 10% data size are respectively 95.78%, 98.33% and 91.55%. Our model achieves the best results among all methods. In contrast, traditional supervised methods (*e.g.* BIND, DF, FS-Net) show substantial F1 performance degradation when the sample size is reduced, *e.g.* Deeppacket's performance decreases by 40.22% when the sample size is reduced from the full size to 10%. This indicates that the pre-training approach solves the classification problem for the few-shot encrypted traffic more effectively.

5 DISCUSSION

In this section, we discuss some limitations of this work, as well as the potential implications it may have in inducing further research in the field. **Generalizability:** The variability of encrypted traffic due to changes in the content of Internet services [13] over time will challenge the ability of our approach with fixed patterns leaned from fixed data and keeping unchanged over time. As the use and rise of TLS 1.3, the labeling of encrypted traffic will not be possible through SNI. We mitigate this in two ways to accommodate the ECH mechanism and thus guarantee the test of generalizability, including active visiting and labeling with unique process identifiers. **Pre-training Security:** Although ET-BERT has good robustness and generalization under a variety of encrypted traffic scenarios, it depends on the clean pre-training data. When an attacker deliberately adds low-frequency subwords as the "toxic" embeddings, a poisoned pre-trained model with a "backdoor" can be generated to force the model to predict the target class and finally fool the normally fine-tuned model on specific classification tasks [14, 15]. However, how to construct the "toxic" tokens of encrypted traffic has not been studied yet.

6 CONCLUSION

In this paper, we propose a new encrypted traffic representation model, ET-BERT, which can pre-train deep contextual datagram-level traffic representations from large-scale unlabeled data, then accurately classify encrypted traffic for multiple scenarios with a simple fine-tuning on a small amount of task-specific labeled data. We comprehensively evaluate the generalization and robustness of ET-BERT on 5 publicly available datasets and the TLS 1.3 dataset collected from CSTNET. ET-BERT has great generalization ability and achieves a new state-of-the-art performance over 5 encrypted traffic classification tasks, including General Encrypted Application Classification, Encrypted Malware Classification, Encrypted Traffic Classification on VPN, Encrypted Application Classification on Tor, Encrypted Application Classification on TLS 1.3, and outperforms existing works remarkably by 5.4%, 0.2%, 5.2%, 4.4%, 10.0%. In the future, we would like to investigate the ability of ET-BERT to predict new classes of samples and to resist sample attacks.

ACKNOWLEDGMENTS

This work is supported by The National Key Research and Development Program of China No. 2021YFB3101400 and the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDC02040400. We are grateful to anonymous reviewers for their fruitful comments, corrections and inspiration to improve this paper. We also sincerely appreciate the shepherding from Magnus Almgren and writing help from Zhong Guan and Lulin Wang.

REFERENCES

[1] Khaled Al-Naami, Swarup Chandra, Ahmad M. Mustafa, Latifur Khan, Zhiqiang Lin, Kevin W. Hamlen, and Bhavani M. Thuraisingham. 2016. Adaptive Encrypted Traffic Fingerprinting with Bi-Directional Dependence. In *the Annual Conference on Computer Security Applications*. 177–188.

[2] Emily Alsentzer, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew B. A. McDermott. 2019. Publicly Available Clinical BERT Embeddings. In *the Clinical Natural Language Processing Workshop*. 72–78.

[3] Amazon. 2021. Alexa Top Sites. https://www.alexa.com/topsites/.

[4] Tomasz Bujlow, Valentín Carela-Español, and Pere Barlet-Ros. 2015. Independent Comparison of Popular DPI Tools for Traffic Classification. *Computer Networks* 76 (2015), 75–89.

[5] Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory (2. ed.)*.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186.

[7] Ali Doganaksoy, Baris Ege, Onur Koçak, and Fatih Sulak. 2010. Cryptographic Randomness Testing of Block Ciphers and Hash Functions. *Cryptology ePrint Archive* (2010), 564.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

[9] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. 2016. Characterization of Encrypted and VPN Traffic using Time-related Features. In *the International Conference on Information Systems Security and Privacy*. 407–414.

[10] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. 2017. Characterization of Tor Traffic using Time based Features. In *the International Conference on Information Systems Security and Privacy*. 253–262.

[11] Jamie Hayes and George Danezis. 2016. K-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *USENIX Security Symposium*. 1187–1203.

[12] Hong Ye He, Zhi Guo Yang, and Xiang Ning Chen. 2020. PERT: Payload Encoding Representation from Transformer for Encrypted Traffic Classification. In *ITU Kaleidoscopef: Industry-Driven Digital Transformation*. 1–8.

[13] Marc Juárez, Sadia Afroz, Gunes Acar, Claudia Díaz, and Rachel Greenstadt. 2014. A Critical Evaluation of Website Fingerprinting Attacks. In *the ACM SIGSAC Conference on Computer and Communications Security*. 263–274.

[14] Nora Kassner and Hinrich Schütze. 2020. Negated and Misprimed Probes for Pretrained Language Models: Birds Can Talk, But Cannot Fly. In *the Annual Meeting of the Association for Computational Linguistics*. 7811–7818.

[15] Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight Poisoning Attacks on Pretrained Models. In *the Annual Meeting of the Association for Computational Linguistics*. 2793–2806.

[16] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations*.

[17] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: A Pre-trained Biomedical Language Representation Model for Biomedical Text Mining. *Bioinformatics* 36, 4 (2019), 1234–1240.

[18] Rui Li, Xi Xiao, Shiguang Ni, Haitao Zheng, and Shutao Xia. 2018. Byte Segment Neural Network for Network Traffic Classification. In *International Symposium on Quality of Service*. 1–10.

[19] Kunda Lin, Xiaolong Xu, and Honghao Gao. 2021. TSCRNN: A Novel Classification Scheme of Encrypted Traffic Based on Flow Spatiotemporal Features for Efficient Management of IIoT. *Computer Networks* 190 (2021), 107974.

[20] Chang Liu, Longtao He, Gang Xiong, Zigang Cao, and Zhen Li. 2019. FS-Net: A Flow Sequence Network For Encrypted Traffic Classification. In *IEEE Conference on Computer Communications*. 1171–1179.

[21] Chuan Liu, Wenyong Wang, Meng Wang, Fengmao Lv, and Martin Konan. 2017. An Efficient Instance Selection Algorithm to Reconstruct Training Set for Support Vector Machine. *Knowledge-Based Systems* 116 (2017), 58–73.

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* (2019).

[23] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. 2020. Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning. *Soft Computing* 24 (2020), 1999–2012.

[24] National Institute of Standards and Technology. 2016. Random Bit Generation: Guide to the Statistical Tests. https://csrc.nist.gov/Projects/Random-Bit-Generation/Documentation-and-Software/Guide-to-the-Statistical-Tests.

[25] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. 2016. Website Fingerprinting at Internet Scale. In *Annual Network and Distributed System Security Symposium*.

[26] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. 2011. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *the Annual ACM Workshop on Privacy in the Electronic Society*. 103–114.

[27] Shahbaz Rezaei and Xin Liu. 2019. Deep Learning for Encrypted Traffic Classification: An Overview. *IEEE Communications Magazine* 57, 5 (2019), 76–81.

[28] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, A Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv preprint arXiv:1910.01108* (2019).

[29] Satadal Sengupta, Niloy Ganguly, Pradipta De, and Sandip Chakraborty. 2019. Exploiting Diversity in Android TLS Implementations for Mobile App Traffic Classification. In *The World Wide Web Conference*. 1657–1668.

[30] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *the International Conference on Information Systems Security and Privacy*. 108–116.

[31] Meng Shen, Jinpeng Zhang, Liehuang Zhu, Ke Xu, and Xiaojiang Du. 2021. Accurate Decentralized Application Identification via Encrypted Traffic Analysis Using Graph Neural Networks. *IEEE Transactions on Information Forensics and Security* 16 (2021), 2367–2380.

[32] Hongtao Shi, Hongping Li, Dan Zhang, Chaqiu Cheng, and Xuanxuan Cao. 2018. An Efficient Feature Generation Approach Based on Deep Learning and Feature Selection Techniques for Traffic Classification. *Computer Networks* 132 (2018), 81–98.

[33] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *the ACM SIGSAC Conference on Computer and Communications Security*. 1928–1943.

[34] Vincent F. Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. 2018. Robust Smartphone App Identification via Encrypted Network Traffic Analysis. *IEEE Transactions on Information Forensics and Security* 13, 1 (2018), 63–78.

[35] Thijs van Ede, Riccardo Bortolameotti, Andrea Continella, Jingjing Ren, Daniel J. Dubois, Martina Lindorfer, David R. Choffnes, Maarten van Steen, and Andreas Peter. 2020. FlowPrint: Semi-Supervised Mobile-App Fingerprinting on Encrypted Network Traffic. In *Annual Network and Distributed System Security Symposium*.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *the International Conference on Neural Information Processing Systems*. 6000–6010.

[37] Kailong Wang, Junzhe Zhang, Guangdong Bai, Ryan K. L. Ko, and Jin Song Dong. 2021. It's Not Just the Site, It's the Contents: Intra-domain Fingerprinting Social Media Websites Through CDN Bursts. In *The Web Conference*. 2142–2153.

[38] Pan Wang, Shuhang Li, Feng Ye, Zixuan Wang, and Moxuan Zhang. 2020. PacketCGAN: Exploratory Study of Class Imbalance for Encrypted Traffic Classification Using CGAN. In *IEEE International Conference on Communications*. 1–7.

[39] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. 2017. Malware Traffic Classification Using Convolutional Neural Network for Representation Learning. In *International Conference on Information Networking*. 712–717.

[40] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. In *the Conference of the Association for Computational Linguistics*. 1441–1451.

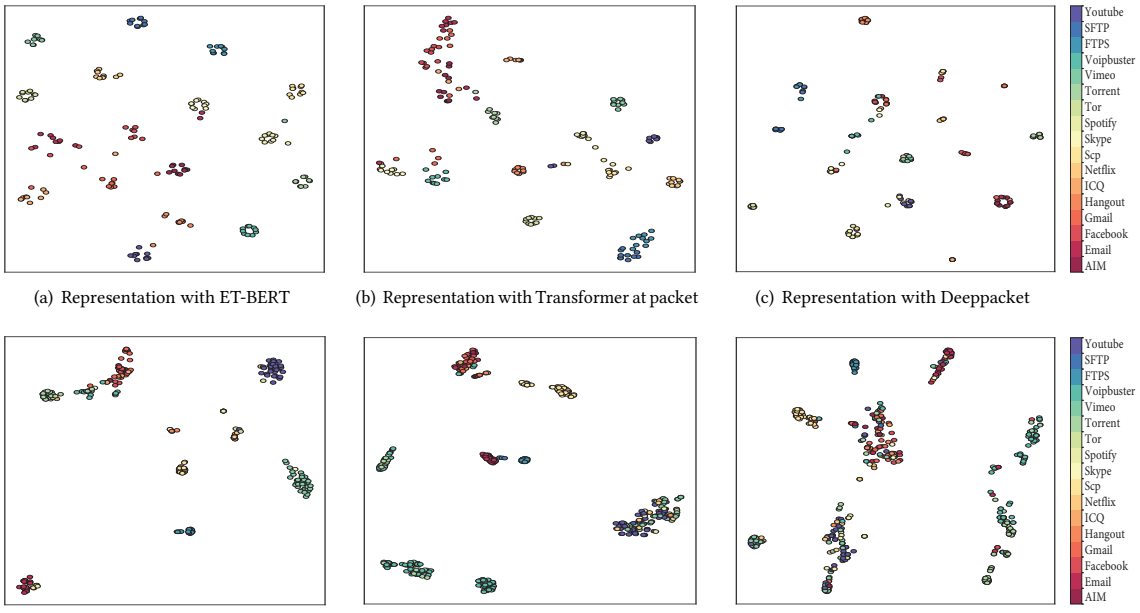
[41] Zhe Zhao, Hui Chen, Jinbin Zhang, Xin Zhao, Tao Liu, Wei Lu, Xi Chen, Haotang Deng, Qi Ju, and Xiaoyong Du. 2019. UER: An Open-Source Toolkit for Pre-training Models. In *the Conference on Empirical Methods in Natural Language Processing*. 241–246.

[42] Wenbo Zheng, Chao Gou, Lan Yan, and Shaocong Mo. 2020. Learning to Classify: A Flow-Based Relation Network for Encrypted Traffic Classification. In *The Web Conference*. 13–22.

A ADDITIONAL COMPARISON STUDY

A.1 Qualitative Analysis

To further evaluate the performance differences between the models, we select 5 models for comparative analysis with ET-BERT, including Transformer at flow level, Transformer at packet level, DF, Deeppacket and PERT. The Transformer model as the baseline of ET-BERT can visually compare the improvement of our pre-trained model, and the remaining three models are representative methods to compare the prominence of our models.



(a) Representation with ET-BERT (b) Representation with Transformer at packet (c) Representation with Deeppacket

(e) Representation with PERT (f) Representation with Transformer at flow (g) Representation with DF

Figure 5: t-SNE Visualization of Classification Boundaries with 6 Methods on ISCX-VPN-App Testset.

We use t-distributed stochastic neighbor embedding (t-SNE) to downscale the test set samples predicted by each model and plot them as two-dimensional images, as in Figure 5. We choose the sampled ISCX-VPN-App dataset in Section 4.3 and then show the best results for each model: (a-c) are packet-level results and (e-g) are flow-level results.

There is no doubt that our model exhibits the best classification performance because ET-BERT captures patterns that can distinguish between different encrypted traffic even under the more secure new encryption protocols. Also with ET-BERT at the packet level, (b) and (c) fail to accurately classify applications especially AIM, ICQ and Gmail, which are used for online chat and Gmail provides online chat service in addition to email service. At the flow level, the classification effect of (f) and (g) is confusing, as YouTube and other streaming applications including Vimeo, Netflix and Spotify cannot be distinguished by these methods, while PERT performs relatively better but still suffers from the interference of applications with the same services.