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

Xinjie Lin¹,², Gang Xiong¹,², Gaopeng Gou¹,², Zhen Li¹,², Junzheng Shi¹, Jing Yu¹,²*  
¹Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China  
²School 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 Encrypted Traffic Bidirectional Encoder Representations from Transformer (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 Lin¹,², Gang Xiong¹,², Gaopeng Gou¹,², Zhen Li¹,², Junzheng Shi¹, Jing Yu¹,²*. 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, 11 pages. https://doi.org/10.1145/3485447.3512217

* Corresponding Author.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

WWW ‘22, April 25–29, 2022, Lyon, France  
© 2022 Copyright held by the owner/author(s).  
ACM ISBN 978-1-4503-9096-5/22/04.  
https://doi.org/10.1145/3485447.3512217

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, 34]. Recently, traffic encryption has been widely utilized to protect the privacy and anonymity 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 [29]. 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 [37] 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
We propose a pre-training framework for encrypted traffic classification tasks, including General Encrypted Application Classification, Encrypted Malware Classification, and Encrypted Traffic classification. Our General ET-BERT model achieves a new state-of-the-art performance over 5 encrypted traffic representations. ET-BERT has great generalization ability and captures both byte-level and self-supervised pre-training tasks, e.g. traffic classification tasks. We propose two traffic-specific pre-training tasks to learn generic traffic representations from large-scale unlabeled encrypted traffic (Figure 1(d)).

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, our model does not rely on any plain-text information. In encrypted traffic classification, although payloads have no semantics, Sengupta et al. [31] 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 [13] 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 [38]. 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 split-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 [28, 33], 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, Token2Embedding 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 \(\text{Trace} = \{\text{flow}_i, i \in \mathbb{N}^+\}\), where \(\text{flow} = \{p_j, j \in \mathbb{N}^+\}\) 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:

\[
\text{BURST} = \begin{cases}
\text{src} = \{p^rc_m, m \in \mathbb{N}^+\} \\
\text{dst} = \{p^ds_n, n \in \mathbb{N}^+\}
\end{cases}
\]  

(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 tokens [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\(^B\).
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 $E_{token}$. The final hidden vector of the input token as $E_{token} \in \mathbb{R}^H$, 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 $E_{pos} \in \mathbb{R}^H$, 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 $E_{seq} \in \mathbb{R}^H$, 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:

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

where $\theta$ represents the set of trainable parameters of ET-BERT. The probability $P$ is modeled by the Transformer encoder with $\theta$. $X$ 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 [39] 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 $B_j = (sub-B_j^A, sub-B_j^B)$ and its ground-truth label $y_j \in [0, 1]$ (0 represents paired sub-BURSTs and 1 represents unpaired ones).

$$L_{SBP} = -\sum_{j=1}^{n} \log(P(y_j|B_j; \theta))$$
Overall, the final pre-training objective is the sum of the above two losses, which is defined as:

$$L = L_{MBP} + L_{SBP}$$  \hfill (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, 32]; (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] representation 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

**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.

| Task | Dataset | #Flow | #Packet | #Label |
|------|---------|-------|---------|--------|
| GEAC | Cross-Platform(iOS) [37] | 20,858 | 707,717 | 196 |
| | Cross-Platform(Android) [37] | 27,846 | 656,044 | 215 |
| EMC | USTC-TFC [41] | 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-L.3 | CSTNET-TLS 1.3 (Ours) | 46,372 | 581,709 | 120 |

**Task 1:** General Encrypted Application Classification (GEAC) task aims to classify application traffic under standard encryption protocols. We test on Cross-Platform (iOS) [37] and Cross-Platform (Android) [37], 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 Malware Classification (EMC) is a collection of encrypted traffic consisting of malware and benign applications [41]. The dataset USTC-TFC [41] contains 10 categories of benign traffic and 10 categories of malicious traffic.

**Task 3:** Encrypted Traffic Classification 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 Application Classification 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 Application Classification on TLS 1.3 (EAC-L.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.

| Method               | AC   | PR   | RC   | F1   | ISCX-VPN-Service | ISCX-VPN-App |
|----------------------|------|------|------|------|------------------|--------------|
| ET-BERT(packet)      | 0.910 | 0.9755 | 0.9772 | 0.9754 | 0.9890 | 0.9891 | 0.9890 | 0.9890 | 0.9962 | 0.9936 | 0.9938 | 0.9937 |
| ET-BERT(flow)        | 0.9044 | 0.9701 | 0.9632 | 0.9643 | 0.9865 | 0.9324 | 0.9266 | 0.9246 | 0.9729 | 0.9756 | 0.9731 | 0.9733 |

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

| Method               | AC   | PR   | RC   | F1   | ISCX-Tor | USTC-TFC | CSTNET-TLS 1.3 |
|----------------------|------|------|------|------|----------|----------|----------------|
| ET-BERT(packet)      | 0.9921 | 0.9923 | 0.9924 | 0.9921 | 0.9929 | 0.9930 | 0.9930 | 0.9930 | 0.9510 | 0.9460 | 0.9419 | 0.9426 |
| ET-BERT(flow)        | 0.8311 | 0.5564 | 0.6448 | 0.5868 | 0.9929 | 0.9930 | 0.9930 | 0.9930 | 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 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 [19, 25, 40], 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 [37, 45]. Macro Average [23] 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 $10^{-5}$ 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 $6 \times 10^{-5}$ for flow-level, and $2 \times 10^{-5}$ 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 [44], 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 [37]; (2) statistical feature methods: AppScanner [36], CUMUL [27], BIND [1] and k-fingerprinting (K-fp) [12]; (3) deep learning methods: Deep Fingerprinting (DF) [35], FS-Net [22], GraphDApp [33], TSCRNN [21], Deeppacket [25]; (4) pre-training method: PERT [13]. 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 to adapt to different platforms, which makes it more practical for real-world deployment.
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.

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.

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

| 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 |

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 [13] 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 [13]. 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.

4.4 Interpretabity

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 valid.
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 [26], where \( p \)-value

\[
\begin{array}{c|cccc}
\text{Cipher Ratio (\%)} & \text{ALL} & 40\% & 20\% & 10\% \\
\hline
\text{Cross-Platform(IOS)} & 100 & 95.78 & 98.33 & 91.55 \\
\text{Cross-Platform(Android)} & 100 & 95.78 & 98.33 & 91.55 \\
\text{USTC} & 100 & 99.14 & 99.21 & 99.30 \\
\text{ISCX-VPN} & 100 & 99.14 & 99.21 & 99.30 \\
\text{ISCX-Tor} & 100 & 99.14 & 99.21 & 99.30 \\
\text{TLS 1.3} & 100 & 99.14 & 99.21 & 99.30 \\
\end{array}
\]

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 [14] 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 [15, 16]. 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.
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.

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