ET-SSM: Linear-Time Encrypted Traffic Classification Method Based On Structured State Space Model

Yanjun Li*, Xiangyu Zhao*, Zhengpeng Zha*†, Zhenhua Ling‡
* Institute of Advanced Technology, University of Science and Technology of China
Email: {ballinli, xyzhao23}@mail.ustc.edu.cn,
† Corresponding author. Email: zhazp@ustc.edu.cn,
‡ School of Information Science and Technology, University of Science and Technology of China
E-mail: zhling@ustc.edu.cn

***Abstract*—Encrypted traffic classification involves categorizing data traffic that has been encrypted for privacy and security. In this domain, Transformers have been essential for successful pretraining. However, their quadratic time complexity makes them costly and slow for inference. While many works tried to reduce memory overhead and speed up inference, they often fail to match the classification performance of existing methods. To tackle this challenge, we present an Encrypted Traffic State Space Model (ET-SSM), utilizing recent advances in sequence routing based on state space models. The ET-SSM initially integrates SSM layers with a multiplicative gated architecture, which has proven effective in simplified sequence modeling frameworks. This structure learns static linear layers that do not account for pairwise traffic interactions. To overcome this limitation, we employ convolution layers to capture adjacent traffic context interactions while maintaining overall linear time complexity. Extensive experiments demonstrate that ET-SSM not only significantly reduces memory overhead and accelerates inference time but also achieves state-of-the-art classification results. This makes it a promising candidate for the pretraining model in encrypted traffic classification for realistic production environments. The source code will be released upon paper acceptance.**

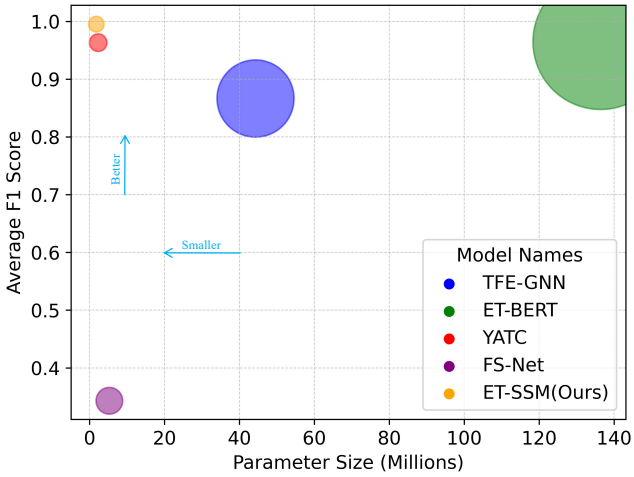


Fig. 1. **The average F1 Score *vs*. parameter size (millions) comparison.** Our ET-SSM achieves the state-of-the-art average F1-score on six encrypted traffic classification datasets while having the smallest parameter size (i.e. The size of the bubbles is depicted in the figure.), compared with cutting-edge encrypted traffic classification model, *e.g.* TFE-GNN [3], ET-BERT [4], YAYC [5], FS-Net [6].

I. Introduction

Recently, the widespread use of traffic encryption has become instrumental in protecting the privacy and anonymity of Internet users [1], [2]. While this advancement is vital for security and confidentiality, it concurrently presents significant challenges to traffic classification. The increasing utilization of privacy-enhancing encryption techniques, such as Tor and VPNs, by both legitimate users and malicious actors, complicates the task of distinguishing benign from harmful traffic. Encrypted traffic classification thus emerges as a crucial tool in this landscape. It enables the identification and mitigation of malware and cybercriminal activities that exploit encryption to bypass surveillance systems, without compromising the privacy and integrity of legitimate communications. This delicate balance between user privacy and cybersecurity underscores the indispensable role of sophisticated encrypted traffic classification methodologies in maintaining a secure digital environment.

In this area, Transformers [7] have become the *de facto* architecture for pretraining models in encrypted traffic classification. Since the introduction of ET-BERT [4], Transformers have proven central to encrypted traffic classification due to their ability to learn effectively from large unlabeled datasets. Specifically, using attention mechanisms [8] as a core routing component has been critical to achieving empirical success in downstream encrypted traffic classification tasks. Moreover, YaTC introduces a masked autoencoder-based traffic transformer [9] with multi-level flow representation to learn more efficient encrypted traffic representations. However, despite the significant success of these attention-based models in encrypted traffic classification, their high memory cost and slow inference time make them challenging to deploy in realistic production environments [1]. In the field of cybersecurity, where encrypted traffic classification is a crucial downstream task, there is a greater emphasis on achieving high throughput rates at a low cost [10]–[12]. Some efforts address the high memory costs and slow inference throughput of pretrain-

ing models for encrypted traffic classification by employing knowledge distillation methods. For instance, XENTC [13] improved the DistilBERT [14] leverages knowledge distillation to enable the use of a compressed model while maintaining good performance. Subsequently, a streamlined packet structure, consisting of a header and a partial payload, is employed to achieve effective feature reduction. Despite these efforts to create lightweight Transformers for encrypted traffic classification, they cannot achieve comparable classification performance. Given the quadratic computational complexity of Transformer models, which escalates with the sequence length, the Transformers cannot balance the effectiveness and efficiency. **Are Transformers the best solution to effective and efficient encrypted traffic classification?**

State Space Models (SSMs) for deep learning offer a promising alternative for sequence modeling. Recent research indicates that SSMs are a competitive architecture for handling sequences, as demonstrated in [15]. Various adaptations of SSMs have demonstrated success across diverse domains. They effectively handle continuous signals, such as audio [16] and visual data [17], as well as discrete signals, such as textual content [18]. SASHIMI [16] is a new architecture for modeling waveforms that yield state-of-the-art performance on conditional audio generation benchmarks. S4ND [17] shows SSMs modeling large-scale visual data across 1D, 2D, and 3D formats as continuous multidimensional signals. Mamba [18] is a promising language model that not only outperforms Transformers of equivalent size but also matches the performance of Transformers twice its size. In addition to delivering strong performance, SSM-based routing avoids the quadratic complexity associated with increasing sequence lengths. Specifically, the model facilitates the capture of recurrent neural network (RNN) long-range dependencies while maintaining convolutional neural networks (CNN) training speeds. Compared to the sparse information density typical of visual data and the denser information density found in speech, text-based SSMs [18] more closely resemble encrypted traffic. However, encrypted traffic classification requires lower computational costs and higher inference speeds compared to the text domain to ensure privacy and anonymity.

The encrypted traffic classification field urgently requires a model that combines low cost, rapid processing speeds, and high classification performance. In this work, we propose an architecture, ET-SSM for applying SSMs for encrypted traffic classification. Specifically, to enhance the model's representative capacity, we have developed a multiplicative gating architecture [19], [20]. This approach simplifies the routing mechanism while maintaining its effectiveness in modeling necessary interactions. Moreover, to better model the adjacent encrypted traffic context without increasing computational costs, we have incorporated CNN blocks into the ET-SSM framework. As observed in Fig. 1, ET-SSM achieves the state-of-the-art average F1-score on six encrypted traffic classification datasets while having the smallest parameter size. Furthermore, compared to existing methods, ET-SSM improves inference speed while maintaining low GPU memory usage. Our contributions are outlined as follows:

- We propose ET-SSM, the first SSM-based method for encrypted traffic classification. Compared to existing classification methods, our approach achieves the lowest GPU memory cost and the fastest inference speed.
- The ET-SSM achieves average state-of-the-art F1 scores across six encrypted traffic classification datasets, consistently outperforming other models on all these datasets.
- Extensive experiments were conducted across a range of encrypted traffic classification tasks to evaluate our methods, including detailed evaluations such as efficiency analyses, and ablation studies.

II. Methods

A. ET-SSM Model Pre-training

Recent work has demonstrated that gating can enhance the performance of models with simplified routing. For instance, linear time attention models have been shown to benefit from improved gating mechanisms [21]. Inspired by the work of [21], we incorporate gated units as the core structural components of our model. The complete ET-SSM model is shown in Figure 2. In the initial phase of our study, we utilize the BURST structure, identified as a sequence of temporally contiguous network packets emanating from either a request or a response in a single session flow. This structure is employed to precisely depict encrypted traffic, thereby forming the input for our pre-trained ET-BERT model, mirroring the strategy delineated in [4]. Subsequently, we combine token embedding, position embedding, and segment embedding to obtain the final encrypted traffic token representation, which serves as the input to the ET-SSM encoder. The final MLP classification head processes the output from the ET-SSM encoder to determine the encrypted traffic categories.

The architecture of ET-SSM is a bidirectional adaptation of the gated unit of [21]. Specifically, let $\mathbf{X}_{i-1} \in \mathbb{R}^{L\times d}$ be activations at the $i-1$-th layer where the length is L, and the model size is d. We use the activation GELU [22] for σ. The ET-SSM layer begins as follows,

$$\begin{aligned}
\mathbf{X}'_{i-1} &= \text{LayerNorm}(\mathbf{X}_{i-1}) && \in \mathbb{R}^{L\times d},\\
\mathbf{Z} &= \sigma(CONV(\mathbf{X_{i-1}})) && \in \mathbb{R}^{L\times 3d},\\
\mathbf{F} &= \sigma(CONV(\mathbf{X}'_{i-1})) && \in \mathbb{R}^{L\times d},\\
\mathbf{B} &= \sigma(CONV(\text{Flip}(\mathbf{X}'_{i-1})) && \in \mathbb{R}^{L\times d}.
\end{aligned} \tag{1}$$

To emulate the bidirectional information extraction capability of BERT [8], we employ two sequential blocks, namely, a forward and a backward SSM layer, as presented,

$$\begin{aligned}
\mathbf{U}_1 &= CONV(\text{SSM}(\mathbf{F})) && \in \mathbb{R}^{L\times d},\\
\mathbf{U}_2 &= CONV(\text{SSM}(\mathbf{B})) && \in \mathbb{R}^{L\times d},\\
\mathbf{U} &= \sigma(CONV((\mathbf{U}_1 \otimes \text{Flip}(\mathbf{U}_2)))) && \in \mathbb{R}^{L\times 3d}.
\end{aligned} \tag{2}$$

Finally, we utilize a residual connection [23] to avoid the gradient vanishing problem. Specifically, we sum the output

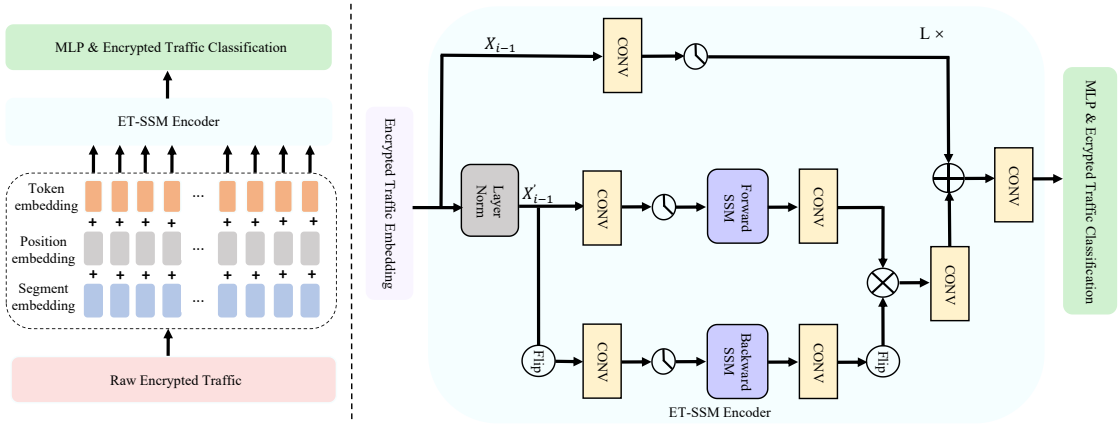


Fig. 2. **The overview of the proposed ET-SSM model.** Initially, we segment the raw encrypted traffic into tokens and incorporate both position and segment embeddings to generate the encrypted traffic embedding, following the approach described by Lin et al. [4]. This embedding is then inputted into our proposed ET-SSM encoder, which processes the sequence bidirectionally. To enhance processing speed and more effectively capture adjacent information, we employ CONV blocks within the encoder. For inference, the MLP classification heads operate following the methodology outlined by Devlin et al. [8], predicting encrypted traffic classifications.

$\mathbf{U}$ with the original input $\mathbf{V}$, as the input $\mathbf{X}_{i+1}$ of the next layer $i+1$.

$$\mathbf{O} = CONV(\mathbf{U} + \mathbf{Z}) \quad \in \mathbb{R}^{L \times d}. \tag{3}$$

However, the multiplicative gated architecture has a significant limitation. It employs static linear layers that are constrained in their ability to account for pairwise traffic interactions. To address this limitation, we incorporate CONV layers to capture adjacent traffic context interactions. By doing so, we enhance the model's ability to understand the relationships between neighboring traffic data points. Importantly, this approach maintains overall linear time complexity, ensuring that the model remains efficient and scalable despite the added complexity of capturing these interactions.

For the feature extraction modules, we use convolutional (CONV) layers instead of fully connected (FC) layers as used in [21] for two reasons. Firstly, the CONV layer is better optimized by compilers than the FC layer [24]. Additionally, With the same number of parameters, a model with CONV 1×1 runs approximately 27% faster than one with FC layers. Finally, suppose FC layers are used as the feature extraction modules. In such cases, additional Permute operations are required to adjust the output feature dimensions to be compatible with the CONV layers, since FC and CONV layers handle different tensor dimensions. However, these Reshape operations may not be hardware-friendly on certain mobile devices [25], [26].

B. ET-SSM Model Fine-tuning

All encoder parameters, including embedding modules and SSM blocks, are loaded from pretraining for downstream tasks. To classify labeled traffic data, the decoder is replaced with an MLP head. Given that all tokens are visible, the fine-tuning of ET-SSM is performed in a supervised manner, as detailed below,

$$\begin{aligned} \mathbf{X} &= \text{Encoder}(\mathbf{X}_0) && \in \mathbb{R}^{L \times D_{\text{enc}}}, \\ \mathbf{f} &= \mathbf{X}[L, :] && \in \mathbb{R}^{D_{\text{enc}}}, \\ \hat{\mathbf{y}} &= \text{MLP}(\text{Norm}(\mathbf{f})). \end{aligned} \tag{4}$$

Here, $\mathbf{f}$ denotes the trailing class token, and $\hat{\mathbf{y}} \in \mathbb{R}^C$ represents the prediction distribution, where C is the number of traffic categories. The classification process is then optimized by minimizing the cross-entropy loss between the prediction distribution $\hat{\mathbf{y}}$ and the ground-truth label $\mathbf{y}$ as below,

$$L_{\text{cls}} = \text{CrossEntropy}(\hat{\mathbf{y}}, \mathbf{y}). \tag{5}$$

III. Experiments

A. Dataset And Metrics

To validate the efficacy and broad applicability of ET-SSM, we conducted a series of experiments on encrypted traffic classification tasks, utilizing six publicly accessible datasets. Due to page limitations, the details of the datasets are provided in Appendix C of the supplementary materials. We evaluate and compare the performance of our model using four typical metrics: Accuracy (AC), Precision (PR), Recall (RC), and F1 Score (F1), as described in [27], [28].

B. Implementation Details

In our approach, we implemented two distinct learning strategies for the ET-SSM model to adapt to different levels of traffic data granularity, the ET-SSM (flow) and the ET-SSM (packet). Due to page limitations, the details of the implementation details are provided in Appendix C of the supplementary materials.

C. Comparison With Existing Methods

Tables I and II showcase the performance comparison of our framework with existing frameworks. The best results are highlighted with an orange line, and the model with the smallest parameters is shown in bold. Our framework sets a

TABLE I

PERFORMANCE COMPARISON OF DIFFERENT METHODS ON CROSS-PLATFORM(IOS), CROSS-PLATFORM(ANDROID), ICSX-VPN-SERVICE DATASETS.THE BEST RESULTS ARE HIGHLIGHTED WITH AN ORANGE LINE, AND THE MODEL WITH THE SMALLEST PARAMETERS IS SHOWN IN **BOLD**.

Dataset	**Params(M)**	**Cross-Platform(iOS)**				**Cross-Platform(Android)**				**ISCX-VPN-Service**			
Method		AC	PR	RC	F1	AC	PR	RC	F1	AC	PR	RC	F1
AppScanner [29]	-	0.3205	0.2103	0.2173	0.2030	0.3868	0.2523	0.2594	0.2440	0.7182	0.7339	0.7225	0.7197
CUMUL [30]	-	0.2910	0.1917	0.2081	0.1875	0.3525	0.2221	0.2409	0.2189	0.5610	0.5883	0.5676	0.5668
BIND [31]	-	0.3770	0.2566	0.2715	0.2484	0.4728	0.3126	0.3253	0.3026	0.7534	0.7583	0.7488	0.7420
K-fp [32]	-	0.2155	0.2037	0.2069	0.2003	0.2248	0.2113	0.2104	0.2052	0.6430	0.6492	0.6417	0.6395
FlowPrint [27]	-	0.9254	0.9438	0.9254	0.9260	0.8698	0.9007	0.8698	0.8702	0.7962	0.8042	0.7812	0.7820
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
FS-Net [34]	5.3	0.3712	0.2845	0.2754	0.2655	0.4846	0.3544	0.3365	0.3343	0.7205	0.7502	0.7238	0.7131
GraphDApp [35]	-	0.3245	0.2450	0.2392	0.2297	0.4031	0.2842	0.2786	0.2703	0.5977	0.6045	0.6220	0.6036
TSCRNN [36]	-	-	-	-	-	-	-	-	-	-	-	-	-
Deeppacket [37]	-	0.9204	0.8963	0.8872	0.9034	0.8805	0.8004	0.7567	0.8138	0.9329	0.9377	0.9306	0.9321
PERT [38]	-	0.9789	0.9621	0.9611	0.9584	0.9772	0.8628	0.8591	0.8550	0.9352	0.9400	0.9349	0.9368
ET-BERT(flow) [4]	187.4	0.9844	0.9701	0.9632	0.9643	0.9865	0.9324	0.9266	0.9246	0.9729	0.9756	0.9731	0.9733
ET-BERT(packet) [4]	187.4	0.9810	0.9757	0.9772	0.9754	0.9728	0.9439	0.9119	0.9206	0.9890	0.9891	0.9890	0.9890
YaTC[39]	2.3	0.9310	0.9307	0.9310	0.9295	0.9042	0.9081	0.9042	0.9042	-	-	-	-
TFE-GNN	-	0.8241	0.8326	0.8241	0.8130	0.8141	0.8308	0.8141	0.8067	-	-	-	-
Ours(flow)	**1.7**	0.9887	0.9932	0.9866	0.9868	0.9894	0.9571	0.9587	0.9579	0.9814	0.9874	0.9877	0.9875
Ours(packet)	**1.7**	0.9954	0.9997	0.9951	0.9974	0.9977	0.9829	0.9891	0.9860	0.9987	0.9934	0.9947	0.9940

TABLE II

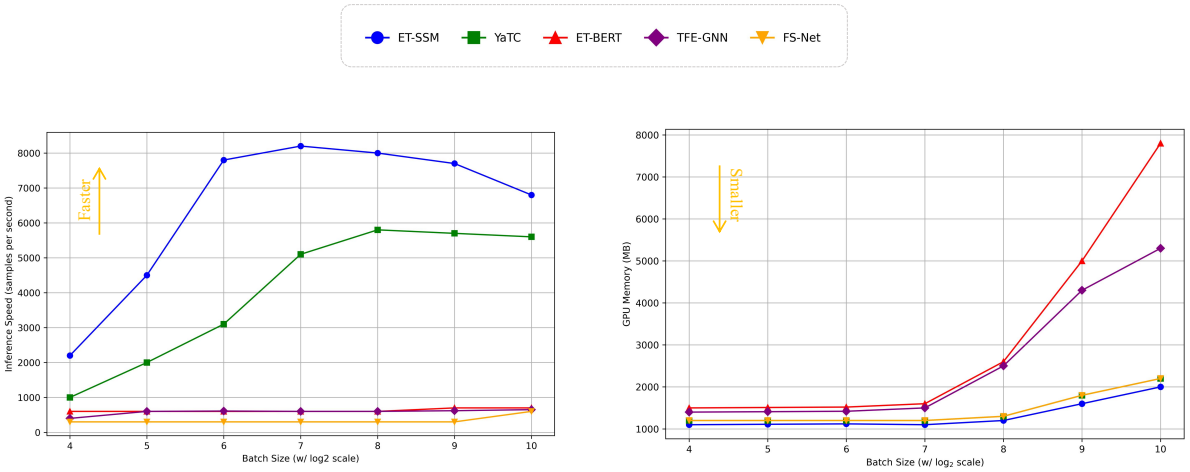
PERFORMANCE COMPARISON OF DIFFERENT METHODS ON ISCX-VPN-APP, ISCX-TOR, USTC-TFC DATASETS.THE BEST RESULTS ARE HIGHLIGHTED WITH AN ORANGE LINE, AND THE MODEL WITH THE SMALLEST PARAMETERS IS SHOWN IN **BOLD**.

Dataset	**Params(M)**	**ISCX-VPN-App**				**ISCX-Tor**				**USTC-TFC**			
Method		AC	PR	RC	F1	AC	PR	RC	F1	AC	PR	RC	F1
AppScanner[29]	-	0.6266	0.4864	0.5198	0.4935	0.6722	0.3756	0.4422	0.3913	0.8954	0.8984	0.8968	0.8892
CUMUL [30]	-	0.5365	0.4129	0.4535	0.4236	0.6606	0.3850	0.4416	0.3918	0.5675	0.6171	0.5738	0.5513
BIND [31]	-	0.6767	0.5152	0.5153	0.4965	0.7185	0.4598	0.4515	0.4511	0.8457	0.8681	0.8382	0.8396
K-fp [32]	-	0.6070	0.5478	0.5430	0.5303	0.6472	0.5576	0.5849	0.5522	-	-	-	-
FlowPrint [27]	-	0.8767	0.6697	0.6651	0.6531	0.9092	0.3820	0.3661	0.3654	0.8146	0.6434	0.7002	0.6573
DF [33]	-	0.6116	0.5706	0.4752	0.4799	0.7533	0.6228	0.6010	0.5850	0.7787	0.7883	0.7819	0.7593
FS-Net [34]	5.3	0.6647	0.4819	0.4848	0.4737	0.6071	0.5080	0.5350	0.4590	0.8846	0.8846	0.8920	0.8840
GraphDApp [35]	-	0.6328	0.5900	0.5472	0.5558	0.6836	0.4864	0.4823	0.4488	0.8789	0.8226	0.8260	0.8234
TSCRNN [36]	-	-	0.9490	0.9480	0.9480	-	0.9870	0.9860	0.9870	-	-	-	-
Deeppacket [37]	-	0.9758	0.9785	0.9745	0.9765	0.7449	0.7549	0.7399	0.7473	0.9640	0.9650	0.9631	0.9641
PERT [38]	-	0.8229	0.7092	0.7173	0.6992	0.7682	0.4424	0.4446	0.4345	0.9909	0.9911	0.9910	0.9911
ET-BERT(flow) [4]	187.4	0.8519	0.7508	0.7294	0.7306	0.8311	0.5564	0.6448	0.5886	0.9929	0.9930	0.9930	0.9930
ET-BERT(packet) [4]	187.4	0.9962	0.9936	0.9938	0.9937	0.9921	0.9923	0.9921	0.9921	0.9915	0.9915	0.9916	0.9916
TFE-GNN	-	-	-	-	-	0.7692	0.8030	0.7692	0.7618	0.9747	0.9747	0.9747	0.9734
YaTC [39]	2.3	0.9819	0.9820	0.9819	0.9819	0.9959	0.9959	0.9959	0.9959	0.9947	0.9749	0.9747	0.9734
Ours(flow)	**1.7**	0.8635	0.7874	0.7597	0.7733	0.8616	0.5876	0.6787	0.6299	0.9934	0.9947	0.9945	0.9946
Ours(packet)	**1.7**	0.9944	0.9978	0.9979	0.9978	0.9969	0.9974	0.9975	0.9976	0.9958	0.9975	0.9976	0.9975

new benchmark in state-of-the-art performance, outperforming the preceding leading method in the F1-score across all six datasets. The margin of enhancement ranges from **+0.16%** to **+6.54%**, as substantiated by the results enumerated in both tables. Moreover, it is imperative to highlight that the results of our experiments surpass all previous state-of-the-art methods in terms of F1-score in the flow-level and packet-level fine-tuning. With state-of-the-art performance in encrypted traffic classification, our ET-SSM model also achieves the smallest parameter count. These outstanding classification results demonstrate that our ET-SSM structure excels in incorporating inductive bias and computational efficiency. For ease of comparison, subsequent experiments and analyses designate ET-SSM(packet) and ET-BERT(packet) as ET-SSM and ET-BERT, respectively.

D. Efficiency Evaluation

Figure 3 presents the experimental analysis of inference speed and GPU memory utilization for the ET-SSM on the largest encrypted traffic classification dataset, Cross-Platform(iOS). As depicted in Figure 3(a), ET-SSM achieved the fastest inference speed among all evaluated methods across various input batch sizes. Specifically, it enhanced inference speed by a factor of 2.2 to 2.5 compared with the previous fastest method YaTC. This significant speed advantage was especially notable given the substantial model parameters and less efficient architectural designs observed in models like ET-BERT, TFE-GNN, and FS-Net. This superior performance highlighted ET-SSM's lower computational complexity than other encrypted traffic classification models. In Figure 3(b), ET-SSM exhibited the lowest GPU memory consumption compared to previous encrypted traffic classification models, even using large batch sizes. Compared to the relatively low-



(a) Inference Speed Comparison

(b) GPU Memory Comparison

Fig. 3. The Inference Speed and GPU Memory Comparison of ET-SSM, YaTC, ET-BERT, TFE-GNN and FS-Net on Cross-Platform(iOS)

cost GPU memory methods FS-Net and YaTC, FS-Net was hindered by significantly slower inference speeds and subpar classification performance. And YaTC was still based on attention structure, hindering its inference speed and GPU memory. Our ET-SSM distinguished itself by significantly enhancing memory efficiency compared to other encrypted traffic classification models.

E. Ablation Studies

Due to page limitations, the details of the ablation studies are provided in Appendix C of the supplementary materials.

IV. Conclusion

In this work, we introduce ET-SSM, a novel framework for encrypted traffic classification designed to elevate both efficiency and effectiveness. This framework integrates SSM layers with a multiplicative gated architecture, which has demonstrated efficacy in simplified sequence modeling contexts. The core architecture of ET-SSM comprises static linear layers that initially do not accommodate pairwise traffic interactions. To address this limitation, we incorporated CNN to capture adjacent traffic context interactions while preserving the overall linear time complexity. Extensive experimental results validate the efficiency and effectiveness of ET-SSM in handling encrypted traffic classification.

Supplementary Materials

V. Appendix A: Related Work

A. State Space Models

State space models (SSMs) [40], [41] have emerged as a promising architecture for sequence modeling. These models synthesize the characteristics of RNNs and CNNs [42], drawing inspiration from classical state space models. SSMs can be implemented efficiently either as a recurrence or convolution, offering linear or near-linear scaling to sequence length. Various adaptations of SSMs have demonstrated success in diverse domains, handling both continuous signals, such as audio [16] and visual [17] data, and discrete signals, such as textual content [18]. SASHIMI [16] is based upon SSMs, resulting in a state-of-the-art performance for both autoregressive and non-autoregressive audio generation. S4ND [17] is an innovative multidimensional SSM layer that extends SSM capabilities to multidimensional data, including images and videos. S4ND excels at modeling large-scale visual data across 1D, 2D, and 3D dimensions as continuous signals. Mamba [18] integrates selective structured state space models into a streamlined, attention-free end-to-end neural network architecture, achieving significant success in the text domain. This model offers rapid inference and linear scaling with sequence lengths, showing enhanced performance on real datasets involving sequences of up to a million elements. These innovative SSMs research efforts inspire us to develop a model based on SSMs that features reduced hyperparameter complexity and accelerated inference speeds for encrypted traffic classification.

B. Encrypted Traffic Classification

Encrypted traffic classification aims to discern the services operating behind obfuscated network traffic, enhancing network service quality and security assurance. Given the remarkable success of the BERT [8] model within the natural language processing community, researchers [4] are exploring the application of its structural principles in the realm of encrypted traffic classification through pre-training approaches. An enhanced traffic classification framework [5] leverages a formatted traffic representation matrix, an efficient Traffic Transformer with hierarchical attention mechanisms, and an MAE-based [9] pre-training paradigm to achieve superior performance with limited labeled data. Although this attention-based [7] pre-training model achieves great classification performance, researchers found that the pre-training model suffers from long processing time and large memory consumption. FastTraffic [43] introduces a novel N-gram feature embedding approach to encapsulate network packet structure and sequential characteristics. Additionally, they design a three-layer MLP that achieves rapid and efficient classification. Furthermore, XENTC [13] enhances the DistilBERT [14] architecture through knowledge distillation, enabling the use of a compressed model while maintaining good performance. Subsequently, they employed a streamlined packet structure, consisting of a header and a partial payload, to achieve effective feature reduction. Despite these two methodologies being less computationally demanding than the attention-based counterparts for encrypted traffic classification, they do not attain comparable levels of performance. What's more, the inherent non-linearity of both the MLP and BERT models, restricts their efficacy in lightweight settings.

Unlike the sparse information density characteristic of visual data and the denser information typical of speech, text-based SSMs [18] more closely resemble encrypted traffic. Nevertheless, encrypted traffic classification demands even lower computational costs and higher inference speeds than those required in the text domain, crucial for ensuring privacy and anonymity. In response, we introduce ET-SSM, a novel framework founded on the principles of SSMs. This approach not only matches state-of-the-art classification performance when compared with existing methods but also reduces GPU memory usage and enhances inference speed. The complete framework of ET-SSM is depicted in Fig. 2.

VI. Appendix B: Methods

A. Preliminaries

A state space model (SSM) is a general-purpose tool for describing the relationship between a continuous-time scalar input $u(t)$ and a scalar output $y(t)$ using the following differential equations,

$$\begin{aligned} x'(t) &= \boldsymbol{A}x(t) + \boldsymbol{B}u(t), \\ y(t) &= \boldsymbol{C}x(t) + \boldsymbol{D}u(t), \end{aligned} \tag{6}$$

where $x(t) \in \mathbb{R}^N$ is a continuous-time state vector, $x'(t)$ is its derivative, and the equation is parameterized by $\boldsymbol{A} \in \mathbb{R}^{N\times N}, \boldsymbol{B} \in \mathbb{R}^{N\times 1}, \boldsymbol{C} \in \mathbb{R}^{1\times N}, \boldsymbol{D} \in \mathbb{R}^{1\times 1}$.

When applied to a discrete-time scalar input sequence $u_1, \dots u_L$, the SSM equations, and parameters can be discretized, leading to the following recursion,

$$x_k = \overline{\boldsymbol{A}}x_{k-1} + \overline{\boldsymbol{B}}u_k, \quad y_k = \overline{\boldsymbol{C}}x_k + \overline{\boldsymbol{D}}u_k, \tag{7}$$

where $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}, \overline{\boldsymbol{D}}$ are functions of the original parameters and a discretization rate.

This equation can be computed similarly to an RNN, where $x_k \in \mathbb{R}^N$ is a hidden state at time step k. However, unlike an RNN, the linearity of the recursion allows $y_1 \dots y_L$ to be computed directly using a convolution with precomputed kernel $\overline{\boldsymbol{K}} \in \mathbb{R}^L$,

$$\begin{aligned} \overline{\boldsymbol{K}} &= (\overline{\boldsymbol{C}}\overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}, \dots, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{L-1}\overline{\boldsymbol{B}}), \\ y &= \overline{\boldsymbol{K}} * u. \end{aligned} \tag{8}$$

An effective approach for integrating SSMs into neural networks has been demonstrated by the works of [15], [41]. Their core insight is the parameterization of the transition matrix $\boldsymbol{A}$, known as HiPPO,

$$\boldsymbol{A}_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k, \\ n+1 & \text{if } n = k, \\ 0 & \text{if } n < k. \end{cases} \tag{9}$$

This matrix yields a stable and efficient training regime. The full model, S4, retains the SSM's ability to model long-term sequences while being more training-efficient than RNNs.

Recently, researchers [44], [45] have proposed simplified diagonalized versions of S4, which achieve comparable results with a more straightforward approximation of the original parameterization. In our preliminary experiments, we tested several different S4 parameterizations and found no significant difference in accuracy. Therefore, we use S4D as the parameterization throughout this work.

B. The Forward Pass Algorithm Description

The operational process of the ET-SSM block forward pass is detailed in Algorithm 1. For a given input token sequence $\mathbf{X}_{i-1}$ with a batch size B and sequence length L from the $(i-1)$-th ET-SSM block, we begin by normalizing it and then projecting it linearly into $\mathbf{x}$ and $\mathbf{z}$, both with a dimension size of E. We subsequently apply causal 1-D convolution to $\mathbf{x}$, resulting in $\mathbf{x}'$. Based on $\mathbf{x}'$, we compute the input-dependent step size Δ, as well as the projection parameters $\mathbf{B}$ and $\mathbf{C}$ having a dimension size of N. We then discretize $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ using Δ. Following this, we calculate $\mathbf{y}$ employing a hardware-aware State Space Model. Finally, $\mathbf{y}$ is gated by $\mathbf{z}$ and added residually to $\mathbf{X}_{i-1}$, resulting in the output token sequence $\mathbf{X}_i$ for the i-th ET-SSM block.

Algorithm 1 ET-SSM Block Process

Require: token sequence $\mathbf{X}_{i-1} : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Ensure: token sequence $\mathbf{X}_i : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
/* normalize the input sequence $\mathbf{X}'_{i-1}$ */
$\mathbf{X}'_{i-1} : (\mathtt{B}, \mathtt{L}, \mathtt{D}) \leftarrow \mathbf{Norm}(\mathbf{X}_{i-1})$
$\mathbf{x} : (\mathtt{B}, \mathtt{L}, \mathtt{E}) \leftarrow \mathbf{Conv1d}^{\mathbf{x}}(\mathbf{X}'_{i-1})$
$\mathbf{z} : (\mathtt{B}, \mathtt{L}, \mathtt{E}) \leftarrow \mathbf{SiLU}(\mathbf{Conv1d}^{\mathbf{z}}(\mathbf{X}'_{i-1}))$
/* process with different direction */
for o in {forward, backward} **do**
  $\mathbf{x}'_o : (\mathtt{B}, \mathtt{L}, \mathtt{E}) \leftarrow \mathbf{SiLU}(\mathbf{Conv1d}_o(\mathbf{x}))$
  $\mathbf{B}_o : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow \mathbf{Linear}_o^{\mathbf{B}}(\mathbf{x}'_o)$
  $\mathbf{C}_o : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow \mathbf{Linear}_o^{\mathbf{C}}(\mathbf{x}'_o)$
  /* softplus ensures positive $\boldsymbol{\Delta}_o$ */
  $\boldsymbol{\Delta}_o : (\mathtt{B}, \mathtt{L}, \mathtt{E}) \leftarrow \log(1 + \exp(\mathbf{Linear}_o^{\boldsymbol{\Delta}}(\mathbf{x}'_o) + \mathbf{Parameter}_o^{\boldsymbol{\Delta}}))$
  /* shape of $\mathbf{Parameter}_o^{\mathbf{A}}$ is $(\mathtt{E}, \mathtt{N})$ */
  $\overline{\mathbf{A}}_o : (\mathtt{B}, \mathtt{L}, \mathtt{E}, \mathtt{N}) \leftarrow \boldsymbol{\Delta}_o \bigotimes \mathbf{Parameter}_o^{\mathbf{A}}$
  $\overline{\mathbf{B}}_o : (\mathtt{B}, \mathtt{L}, \mathtt{E}, \mathtt{N}) \leftarrow \boldsymbol{\Delta}_o \bigotimes \mathbf{B}_o$
  $\mathbf{y}_o : (\mathtt{B}, \mathtt{L}, \mathtt{E}) \leftarrow \mathbf{SSM}(\overline{\mathbf{A}}_o, \overline{\mathbf{B}}_o, \mathbf{C}_o)(\mathbf{x}'_o)$
end for
/* get gated $\mathbf{y}_o$ */
$\mathbf{y}'_{forward} : (\mathtt{B}, \mathtt{L}, \mathtt{E}) \leftarrow \mathbf{Conv1d}^{\mathbf{forward}}(\mathbf{y}_{forward})$
$\mathbf{y}'_{backward} : (\mathtt{B}, \mathtt{L}, \mathtt{E}) \leftarrow \mathbf{Conv1d}^{\mathbf{backward}}(\mathbf{y}_{backward})$
$\mathbf{y}_{hadamard} : (\mathtt{B}, \mathtt{L}, \mathtt{E}) \leftarrow \mathbf{y}'_{forward} \odot \mathbf{y}'_{backward}$
$\mathbf{y} : (\mathtt{B}, \mathtt{L}, \mathtt{E}) \leftarrow \leftarrow \mathbf{Conv1d}^{\mathbf{y_{hadamard}}}(\mathbf{y}_{hadamard})$
$\mathbf{T}_i : (\mathtt{B}, \mathtt{L}, \mathtt{D}) \leftarrow (\mathbf{z} + \mathbf{y})$
/* residual connection */
Return: $\mathbf{T}_i$

VII. Appendix C: Experiments

A. Dataset And Metrics

To validate the efficacy and broad applicability of ET-SSM, we conducted a series of experiments on encrypted traffic classification tasks, utilizing six publicly accessible datasets. Table III delineates the specifics of these datasets. The General Encrypted Application Classification [27] task categorizes application traffic under standard encryption protocols. Our evaluations were conducted on the Cross-Platform datasets for both iOS and Android, encompassing 196 and 215 applications respectively. The Encrypted Malware Classification [46] task involves the analysis of encrypted traffic comprising both malware and benign applications. In this context, the USTC-TFC dataset is particularly noteworthy, as it features 10 categories each of benign and malicious traffic, providing a comprehensive framework for assessing encryption-based malware detection capabilities. The Encrypted Traffic Classification on VPN [47] task focuses on classifying encrypted traffic that utilizes Virtual Private Networks (VPNs) for network communication. We employ the widely-used ISCX-VPN dataset, which comprises data from six communication applications, captured through the Canadian Institute for Cybersecurity in both VPN and non-VPN settings. The Encrypted Application Classification on Tor [48] task is centered around classifying encrypted traffic using the Onion Router to enhance communication privacy. The relevant dataset, ISCX-Tor, comprises 16 distinct applications, offering a unique landscape for assessing privacy-preserved encrypted traffic analysis.

We evaluate and compare the performance of our model using four typical metrics: Accuracy (AC), Precision (PR), Recall (RC), and F1 Score (F1), as described in [27], [28].

To mitigate the effects of data imbalance across multiple categories, we employ Macro Averaging [49] which calculates the mean value of AC, PR, RC, and F1 for each category.

B. Implementation Details

During the pre-training, approximately 30GB of traffic data was utilized for this procedure. The dataset was divided into two segments: (1) roughly 15GB of traffic data sourced from public datasets [47], [50]; (2) an equivalent volume of traffic data, approximately 15GB, obtained through passive collection within the China Science and Technology Network (CSTNET). Furthermore, we adopted the training data and strategies outlined by Izsak et al. [51]. In line with the approach used by RoBERTa [52], we exclusively employed masked language modeling, preceding next-sentence prediction. The batch size was set at 128, and the total number of steps was 500,000. The learning rate was established at 1×10^{-5}, with a warmup ratio of 0.1. For fine-tuning, we utilized the AdamW optimizer across 10 epochs, applying a learning rate of 6×10^{-5} for flow-level and 2×10^{-5} for packet-level tasks. The batch size remained at 32, and the dropout rate was set to 0.5.

TABLE III
SUMMARY OF DATASETS USED IN ENCRYPTED TRAFFIC CLASSIFICATION EXPERIMENTS

Task	Dataset	Flow	Packet	Label
General Encrypted Application Classification	Cross-Platform (iOS) [27]	20,858	707,717	196
	Cross-Platform (Android) [27]	27,846	656,044	215
Encrypted Malware Classification	USTC-TFC [46]	9,853	97,115	20
Encrypted Traffic Classification on VPN	ISCX-VPN-Service [47]	3,694	60,000	12
	ISCX-VPN-App [47]	2,329	77,163	17
Encrypted Application Classification on Tor	ISCX-Tor [48]	3,021	80,000	16

TABLE IV
COMPARISON OF DIFFERENT METHODS, THEIR CORE STRUCTURES, AND THE COMPLEXITY PER LAYER.

Symbolic method	Core structure	Sequential operations	Complexity per layer
DeepPacket [37]	CNN	$\mathcal{O}(1)$	$\mathcal{O}(k \cdot \mathcal{L} \cdot \mathcal{D}^2)$
TSCRNN [36]	RNN	$\mathcal{O}(\mathcal{L})$	$\mathcal{O}(\mathcal{L} \cdot \mathcal{D}^2)$
FastTraffic [43]	MLP	$\mathcal{O}(1)$	$\mathcal{O}(\mathcal{L} \cdot \mathcal{H})$
ET-BERT [4]	MHA	$\mathcal{O}(1)$	$\mathcal{O}(h \cdot \mathcal{L}^2 \cdot \mathcal{D})$
Ours(ET-SSM)	SSMs	$\mathcal{O}(1)$	$\mathcal{O}(h \cdot \mathcal{L} \cdot \mathcal{D})$

All experiments were conducted using Pytorch 1.8.0 on eight NVIDIA V100 GPUs. In our approach, we implemented two distinct learning strategies for the ET-SSM model to adapt to different levels of traffic data granularity, the ET-SSM (flow) and the ET-SSM (packet).

For testing, we maintained consistency in the dataset across both strategies, ensuring a fair and objective comparison with other methodologies. The pivotal difference between these strategies lay in the granularity of the fine-tuning input traffic information. Our method employed a dataset comprising a concatenated sequence of M consecutive packets within a flow, where M is predefined as 5 in our experimental setup.

C. Efficiency Evaluation

We conducted a theoretical analysis of the time complexity. The computational complexity for each layer was detailed in Table IV. In this table, we assigned a symbolic method to each core architecture type and subsequently analyzed their per-layer time complexities. Our ET-SSM's core architecture was based on State Space Models (SSMs), contrasting with other encrypted traffic classification methods that utilized CNN, RNN, MLP, and Multi-Head Attention (MHA). The parameters defined in the table included $\mathcal{L}$, which represents the input length; k, the kernel size of the convolution; $\mathcal{D}$, the representation dimension, and h, the number of attention heads, where H is typically less than $\mathcal{D}^2$. Consequently, ET-SSM's per-layer computational complexity was notably lower compared to CNN, RNN, MLP, and MHA. Furthermore, MLP, CNN, and MHA support parallel operations through vector multiplication, reducing their sequential operation complexity to merely $\mathcal{O}(1)$. In contrast, RNN requires $\mathcal{O}(\mathcal{L})$, due to its dependence on sequential execution at each time step. This sequential execution analysis shows that SSM-based routing does not exhibit quadratic complexity as the sequence length increases. Specifically, the ET-SSM facilitates achieving RNN-like long-range dependencies with a training speed comparable to CNNs.

D. Ablation Studies

In this study, we evaluated the impact of individual components within our framework at the packet level across six encrypted traffic classification datasets, as detailed in Table V. We established the original ET-SSM as our baseline for comparison. Initially, replacing the bidirectional SSM with a unidirectional SSM resulted in a minimum decline of **-3.93%** in the F1-score for the ISCX-VPN-Service dataset. This decline likely stemmed from the inability of the unidirectional SSM to capture bidirectional information as effectively as architectures like BERT [8] do. Additionally, substituting the convolutional module with a linear projection led to a decrease in the F1 score across all datasets, with a maximum drop of **-1.40%** observed in the Cross-Platform(iOS) dataset. This reduction may be attributed to the linear projection's inferior capability to capture adjacent information, which was crucial for enhancing the model's performance. Lastly, reverting to the original masking and next sentence prediction strategies employed in BERT [8] resulted in a maximum reduction of **-1.28%** in the F1-score for the Cross-Platform(iOS). This outcome underscored the effectiveness of masking strategies, similar to those proposed in [52], in encrypted traffic classification tasks.

E. Different Label Data Size Analysis

We conducted an extensive analysis to assess the impact of varying percentages of fine-tuning dataset volumes, with the variance in F1-score illustrated by the size of the error bands under packet-level fine-tuning, as depicted in Figures 4 and 5. These figures highlight the comparative performance of ET-SSM, YaTC, and ET-BERT across six datasets. It is observed that as dataset volumes decrease, ET-SSM consistently outperforms ET-BERT, achieving a minimum increase of **+0.41%** in classification performance. This improvement becomes even more pronounced at reduced dataset volumes, particularly below 40%, as shown in Figures 4 and 5. Furthermore, the

TABLE V
ABLATION STUDY ON SIX ENCRYPTED TRAFFIC CLASSIFICATION DATASETS

Method	Cross-Platform (iOS)		Cross-Platform (Android)		ISCX-VPN-Service		ISCX-VPN-App		ISCX-Tor		USTC-TFC	
	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1	AC	F1
ET-SSM	0.9954	0.9974	0.9977	0.9860	0.9987	0.9940	0.9944	0.9978	0.9969	0.9976	0.9958	0.9975
w/o Bidirection	0.9571	0.9581	0.9417	0.9401	0.9412	0.9387	0.9391	0.9402	0.9423	0.9487	0.9413	0.9434
w/o Conv	0.9814	0.9834	0.9825	0.9764	0.9841	0.9864	0.9832	0.9864	0.9879	0.9893	0.9874	0.9865
w/o RoBERTa masking	0.9871	0.9857	0.9848	0.9804	0.9854	0.9823	0.9821	0.9879	0.9871	0.9903	0.9846	0.9847

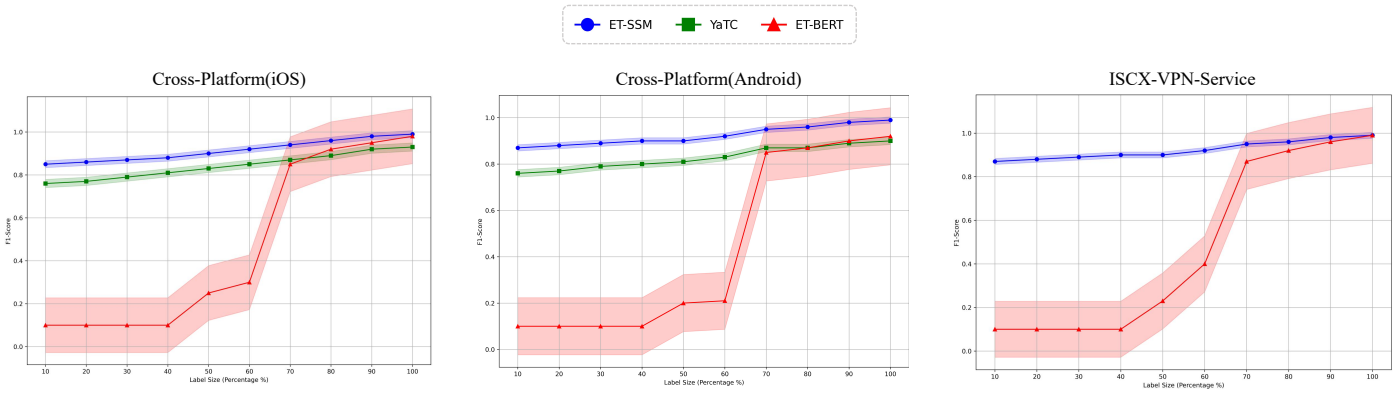


Fig. 4. F1-scores of ET-SSM, YaTC, and ET-BERT using varying percentages of fine-tuning samples on Cross-Platform(iOS), Cross-Platform(Android), and ISCX-VPN-Service datasets

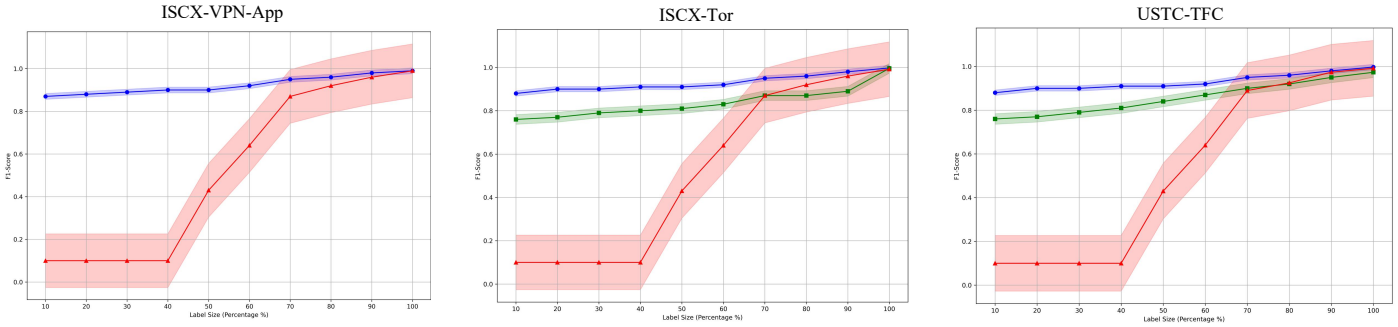


Fig. 5. F1-scores of ET-SSM, YaTC, and ET-BERT using varying percentages of fine-tuning samples on ISCX-VPN-App, ICSX-Tor, and USTC-TFC datasets

error band associated with ET-BERT is broader than that of ET-SSM, indicating greater variance in ET-BERT's F1 scores compared to those of ET-SSM. This observation underscores the superior robustness and enhanced classification efficacy of our proposed framework relative to ET-BERT when applied to packet-level fine-tuning. Conversely, while YaTC demonstrates comparably low variance, akin to ET-SSM, and performs well under limited label data scenarios, it does not match the robust encrypted traffic classification capabilities of ET-SSM across all dataset volumes. These findings further illustrate the superior robustness and improved classification efficacy of our framework compared to YaTC under similar conditions. YaTC did not test performance on the ISCX-VPN-Service and ISVC-VPN-App datasets under varying percentages of fine-tuning samples. Instead, we only compared our method with ET-BERT on these datasets.

REFERENCES

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

[2] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management (IJNM)*, vol. 25, no. 5, pp. 355–374, 2015.

[3] H. Zhang, L. Yu, X. Xiao, *et al.*, "Tfe-gnn: A temporal fusion encoder using graph neural networks for fine-grained encrypted traffic classification," in *Proceedings of the 32nd World Wide Web Conference (WWW)*, ACM, 2023, pp. 2066–2075.

[4] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classifica-

tion," in *Proceedings of the ACM Web Conference 2022 (WWW)*, 2022, pp. 633–642.

[5] R. Zhao, M. Zhan, X. Deng, *et al.*, "Yet another traffic classifier: A masked autoencoder based traffic transformer with multi-level flow representation," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 37, 2023, pp. 5420–5427.

[6] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: A flow sequence network for encrypted traffic classification," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, IEEE, 2019, pp. 1171–1179.

[7] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in Neural Information Processing Systems (NIPS)*, vol. 30, 2017.

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

[9] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 16 000–16 009.

[10] U. Tariq, I. Ahmed, A. K. Bashir, and K. Shaukat, "A critical cybersecurity analysis and future research directions for the internet of things: A comprehensive review," *Sensors*, vol. 23, no. 8, p. 4117, 2023.

[11] A. Handa, A. Sharma, and S. K. Shukla, "Machine learning in cybersecurity: A review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 4, e1306, 2019.

[12] O. Abdulkader, A. M. Bamhdi, V. Thayananthan, F. Elbouraey, and B. Al-Ghamdi, "A lightweight blockchain based cybersecurity for iot environments," in *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, IEEE, 2019, pp. 139–144.

[13] C.-Y. Shin, J.-T. Park, U.-J. Baek, and M.-S. Kim, "A feasible and explainable network traffic classifier utilizing distilbert," *IEEE Access*, 2023.

[14] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[15] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "Hippo: Recurrent memory with optimal polynomial projections," *Advances in Neural Information Orocessing systems (NIPS)*, vol. 33, pp. 1474–1487, 2020.

[16] K. Goel, A. Gu, C. Donahue, and C. Ré, "It's raw! audio generation with state-space models," in *International Conference on Machine Learning (ICML)*, PMLR, 2022, pp. 7616–7633.

[17] E. Nguyen, K. Goel, A. Gu, *et al.*, "S4nd: Modeling images and videos as multidimensional signals using state spaces," *arXiv preprint arXiv:2210.06583*, 2022.

[18] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[19] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *International Conference on Machine Learning (ICML)*, PMLR, 2017, pp. 933–941.

[20] H. Mehta, A. Gupta, A. Cutkosky, and B. Neyshabur, "Long range language modeling via gated state spaces," *arXiv preprint arXiv:2206.13947*, 2022.

[21] W. Hua, Z. Dai, H. Liu, and Q. Le, "Transformer quality in linear time," in *International Conference on Machine Learning (ICML)*, PMLR, 2022, pp. 9099–9117.

[22] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[24] X. Liu, J. Pool, S. Han, and W. J. Dally, "Efficient sparse-winograd convolutional neural networks," *arXiv preprint arXiv:1802.06367*, 2018.

[25] Y. Li, J. Hu, Y. Wen, *et al.*, "Rethinking vision transformers for mobilenet size and speed," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, 2023, pp. 16 889–16 900.

[26] Y. Li, G. Yuan, Y. Wen, *et al.*, "Efficientformer: Vision transformers at mobilenet speed," *Advances in Neural Information Processing Systems (NIPS)*, vol. 35, pp. 12 934–12 949, 2022.

[27] T. Van Ede, R. Bortolameotti, A. Continella, *et al.*, "Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," in *Network and distributed system security symposium (NDSS)*, vol. 27, 2020.

[28] W. Zheng, C. Gou, L. Yan, and S. Mo, "Learning to classify: A flow-based relation network for encrypted traffic classification," in *Proceedings of The Web Conference 2020*, 2020, pp. 13–22.

[29] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 13, no. 1, pp. 63–78, 2017.

[30] A. Panchenko, F. Lanze, J. Pennekamp, *et al.*, "Website fingerprinting at internet scale.," in *NDSS*, 2016.

[31] K. Al-Naami, S. Chandra, A. Mustafa, *et al.*, "Adaptive encrypted traffic fingerprinting with bi-directional dependence," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016, pp. 177–188.

[32] J. Hayes and G. Danezis, "K-fingerprinting: A robust scalable website fingerprinting technique," in *25th USENIX Security Symposium (USENIX)*, 2016, pp. 1187–1203.

[33] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 1928–1943.

[34] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: A flow sequence network for encrypted traffic classification," in *IEEE INFOCOM 2019-IEEE Conference On Computer Communications*, IEEE, 2019, pp. 1171–1179.

[35] M. Shen, J. Zhang, L. Zhu, K. Xu, and X. Du, "Accurate decentralized application identification via encrypted traffic analysis using graph neural networks," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 16, pp. 2367–2380, 2021.

[36] K. Lin, X. Xu, and H. Gao, "Tscrnn: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of iiot," *Computer Networks*, vol. 190, p. 107 974, 2021.

[37] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[38] H. Y. He, Z. G. Yang, and X. N. Chen, "Pert: Payload encoding representation from transformer for encrypted traffic classification," in *2020 ITU Kaleidoscope: Industry-Driven Digital Transformation (ITU K)*, IEEE, 2020, pp. 1–8.

[39] R. Zhao, M. Zhan, X. Deng, *et al.*, "Yet another traffic classifier: A masked autoencoder based traffic transformer with multi-level flow representation," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, AAAI Press, 2023, pp. 5420–5427.

[40] A. Gu, I. Johnson, K. Goel, *et al.*, "Combining recurrent, convolutional, and continuous-time models with linear state space layers," *Advances in Neural Information Processing Systems (NIPS)*, vol. 34, pp. 572–585, 2021.

[41] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," *arXiv preprint arXiv:2111.00396*, 2021.

[42] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[43] Y. Xu, J. Cao, K. Song, Q. Xiang, and G. Cheng, "Fasttraffic: A lightweight method for encrypted traffic fast classification," *Computer Networks*, vol. 235, p. 109 965, 2023.

[44] A. Gu, A. Gupta, K. Goel, and C. Ré, "On the parameterization and initialization of diagonal state space models," *arXiv preprint arXiv:2206.11893*, 2022.

[45] A. Gupta, "Diagonal state spaces are as effective as structured state spaces," *arXiv preprint arXiv:2203.14343*, 2022.

[46] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International conference on information networking (ICOIN)*, IEEE, 2017, pp. 712–717.

[47] G. D. Gil, A. H. Lashkari, M. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related features," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, SciTePress Setúbal, Portugal, 2016, pp. 407–414.

[48] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *International Conference on Information Systems Security and Privacy (ICISSP)*, SciTePress, vol. 2, 2017, pp. 253–262.

[49] C. Liu, W. Wang, M. Wang, F. Lv, and M. Konan, "An efficient instance selection algorithm to reconstruct training set for support vector machine," *Knowledge-Based Systems*, vol. 116, pp. 58–73, 2017.

[50] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," *ICISSP*, vol. 1, pp. 108–116, 2018.

[51] P. Izsak, M. Berchansky, and O. Levy, "How to train bert with an academic budget," *arXiv preprint arXiv:2104.07705*, 2021.

[52] Y. Liu, M. Ott, N. Goyal, *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.