

FlowletFormer: Network Behavioral Semantic Aware Pre-training Model for Traffic Classification

Liming Liu ^{*1}, **Ruoyu Li** ^{*2}, **Qing Li** ³, **Meijia Hou** ⁴, **Yong Jiang** ^{1 3}, **Mingwei Xu** ⁵

¹ Tsinghua Shenzhen International Graduate School ² Shenzhen University

³ Peng Cheng Laboratory ⁴ Zhongguancun Laboratory ⁵ Tsinghua University

Abstract

Network traffic classification using pre-training models has shown promising results, but existing methods struggle to capture packet structural characteristics, flow-level behaviors, hierarchical protocol semantics, and inter-packet contextual relationships. To address these challenges, we propose FlowletFormer, a BERT-based pre-training model specifically designed for network traffic analysis. FlowletFormer introduces a Coherent Behavior-Aware Traffic Representation Model for segmenting traffic into semantically meaningful units, a Protocol Stack Alignment-Based Embedding Layer to capture multilayer protocol semantics, and Field-Specific and Context-Aware Pretraining Tasks to enhance both inter-packet and inter-flow learning. Experimental results demonstrate that FlowletFormer significantly outperforms existing methods in the effectiveness of traffic representation, classification accuracy, and few-shot learning capability. Moreover, by effectively integrating domain-specific network knowledge, FlowletFormer shows better comprehension of the principles of network transmission (e.g., stateful connections of TCP), providing a more robust and trustworthy framework for traffic analysis.

Introduction

Network traffic refers to data transmitted across networks, including the exchange of packets and other forms of communication between devices. It comprises payload and metadata that provide key insights into network behavior. Monitoring and analyzing traffic is essential for both network management and security (Papadogiannaki and Ioannidis 2022; Tang et al. 2020), enabling network operators to effectively tailor resource allocation, guarantee quality of service, and detect malicious activities (Guterman et al. 2019; Hu et al. 2023; Mao et al. 2019).

Traditional rule-based methods for traffic classification (Roesch 1999) struggle with complex, encrypted traffic in modern networks (Taylor et al. 2016). The researchers then turned to machine learning (ML), which proved effective in traffic classification (Al-Naami et al. 2016; Panchenko et al. 2016), although it depends on manual feature engineering, requiring expert knowledge and considerable effort. Deep Learning (DL) methods address these challenges

by automatically extracting features from raw traffic, capturing complex patterns (Sirinam et al. 2018; Liu et al. 2019a; Shen et al. 2021; Schuster, Shmatikov, and Tromer 2017). However, they require large-scale labeled datasets, which are difficult to obtain (Rezaei and Liu 2019; Shen et al. 2020; Aouedi et al. 2022).

Recently, pre-training methods (He, Yang, and Chen 2020; Zhao et al. 2023; Lin et al. 2022; Zhou et al. 2025) have emerged and exhibited superior performance in traffic classification tasks. These methods involve pre-training model on a large volume of unlabeled data to learn general representations, which can subsequently be fine-tuned for specific traffic classification tasks using smaller labeled datasets. Approaches like PERT (He, Yang, and Chen 2020), ET-BERT (Lin et al. 2022), and Trafficformer (Zhou et al. 2025) employing BERT models have demonstrated promising results.

Despite achieving promising accuracy on given datasets, existing pre-training models for traffic classification still exhibit significant limitations.

First, traffic corpora construction often mechanically adopts NLP techniques, such as 4-hex Bigram Encoding and subword tokenization, overlooking packet structure and field semantics. 4-hex Bigram Encoding represents traffic as “words”, which fail to reflect critical 1- to 2-hex fields (e.g., IP version, TTL). Subword tokenization is used, but fewer than 1% of “words” are further segmented.

Second, existing methods fail to model the hierarchical semantics of network protocols, treating packets as flat sequences and ignoring structural differences across layers. For example, the first two bytes at the IP layer and the first two bytes at the TCP layer may have identical HEX values but represent entirely different fields.

Third, existing pretraining tasks are limited in their ability to capture flow-level semantics. Due to 65% of “sentence” containing only a single packet, the SBP tasks in ET-BERT largely degrade into intra-packet association predictions while only 20% support complete traffic behavioral pattern learning.

As a consequence of these limitations, existing approaches are hard to learn network contextual information and general patterns in network traffic. To address these challenges, we propose FlowletFormer, a BERT-based pre-training model for network traffic analysis. Specifically, we

^{*}These authors contributed equally.

make the following contributions:

- 1) To overcome the limitations in existing traffic representation, we propose a coherent behavioral unit, **Flowlet**, as our traffic representation. Flowlets aggregate packets within a logical interaction. To encode them effectively, we further introduce **Field Tokenization**, which converts each flowlet into semantically meaningful tokens based on protocol header fields.
- 2) To address the overlook of the hierarchical structure of network protocols in existing models, we propose a **Protocol Stack Alignment-Based Embedding Layer** that aligns input tokens with the hierarchical structure of protocols. By encoding multi-layer semantics, our method enables the model to distinguish fields across protocol boundaries and better capture protocol-specific behaviors.
- 3) To bridge the gaps in pretraining tasks, we introduce two novel pretraining tasks. The **Masked Field Model** enhances field-level semantic understanding by predicting selectively masked critical protocol fields. The **Flowlet Prediction Task** captures logical interactions by modeling relations between Flowlets, such as HTTP requests and disconnections. By incorporating both intra-packet and inter-packet context, our method captures general traffic patterns.

We evaluate FlowletFormer on 8 fine-tuning datasets, achieving state-of-the-art performance on 7 of them, with over 5% F1 improvement on 5 datasets. To assess the capability of the pre-trained model¹ in understanding the underlying principles of network transmission and protocol header semantics, we novelly propose several **field understanding tasks** and **word analogy similarity analyses**. Our code is available in supplementary material.

Related Work

Traffic Classification

Traffic classification methods have evolved significantly in the past decade, driven by the increasing complexity of network traffic and the need for efficient network management. Early approaches to traffic classification focused primarily on the statistical analysis of packet headers and flow characteristics. For example, early work (Roesch 1999; Zuev and Moore 2005) utilized packet-based, flow-based features or rule matching, such as packet size and inter-arrival times. However, it becomes ineffective in encrypted traffic where observable patterns are largely concealed.

Early ML-based methods, such as those proposed in (Taylor et al. 2016; Al-Naami et al. 2016; Panchenko et al. 2016; Sommer and Paxson 2010), explored classifiers such as decision trees, random forests, and SVMs to classify traffic. These approaches leveraged features including statistical summaries of flow-level metrics and protocol-specific characteristics. Feature engineering requires significant domain expertise and is time-consuming to engineer.

In recent years, deep learning techniques have been increasingly applied to traffic classification, offering superior performance due to their ability to learn complex, high-dimensional representations of network traffic. Lotfollahi

(Lotfollahi et al. 2020) introduced a DNN-based approach that directly uses raw packet data, bypassing feature extraction. This end-to-end approach significantly improved accuracy over other methods. CNNs, RNNs, and GNNs have been widely applied to traffic classification tasks (Sirinam et al. 2018; Liu et al. 2019a; Shen et al. 2021; Schuster, Shmatikov, and Tromer 2017; Zhang et al. 2020). However, their high performance often depends on large-scale labeled datasets, which are expensive to obtain in practice.

Pre-training Methods

Pre-training methods have brought breakthroughs to both NLP and CV by enabling models to learn from large-scale unlabeled data. Early models like ELMo (Peters et al. 2018) used bidirectional LSTMs to capture contextual information. Later, Transformer-based architectures (Vaswani 2017) such as BERT (Devlin et al. 2019) and GPT (Radford et al. 2018) popularized pre-training through tasks like Masked Language Modeling. RoBERTa (Liu et al. 2019b) further improved BERT by removing the Next Sentence Prediction objective and increasing training scale, while ALBERT (Lan et al. 2020) reduced model size through parameter sharing.

Due to its powerful sequence modeling capabilities, the transformer architecture has also been widely applied to network traffic classification tasks (He, Yang, and Chen 2020; Zhao et al. 2023; Lin et al. 2022; Zhou et al. 2025). Recent works have adapted Transformer-based pre-training to network traffic. PERT (He, Yang, and Chen 2020) directly feeds raw packets into a Transformer, while YaTC (Zhao et al. 2023) converts flows into structured images for masked autoencoder training. BERT and TrafficFormer (Lin et al. 2022; Zhou et al. 2025) segment flow into BURST by changes in the direction of packets, which is considered to represent transmission patterns from the application layer. Then each packet in BURST is serialized into a hexadecimal string and represents by 4-hex Bigram Encoding (e.g., `0x45 0x00 0x00 0x34 ...` into `4500, 0000, 0034, ...`). Finally, they apply Subword Tokenization (Chung, Cho, and Bengio 2016; Sennrich, Haddow, and Birch 2016; Luong and Manning 2016) over the bigram tokens to build a fixed-size vocabulary and split tokens (e.g., `1df8` into `1df` and `#8`). The resulting tokens are then used for both pre-training and downstream tasks.

However, these designs based on BURST, 4-hex Bigram Encoding and Subword Tokenization do not fully align with the common patterns of network traffic or the semantic features within packets, which limits the model's understanding of network traffic.

FlowletFormer

FlowletFormer introduces a novel framework that enables the model to capture fine-grained network behaviors and hierarchical semantics in traffic.

Flowlet and Field Tokenization

Current pre-training models repurpose NLP representation and tokenization for network traffic, ignoring their distinct structure and semantics.

¹That is, the model after pretraining but before fine-tuning.

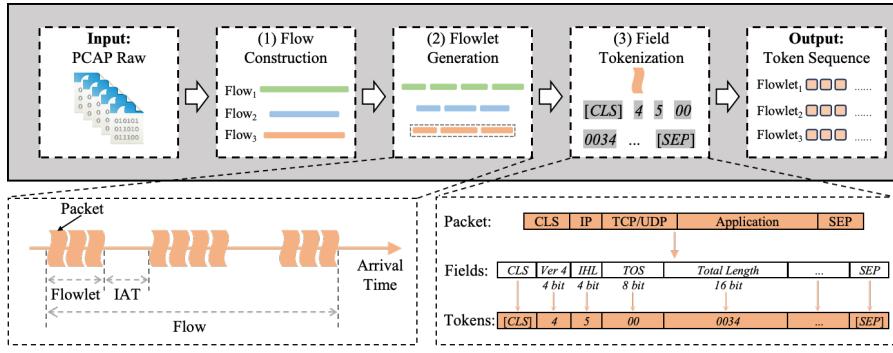


Figure 1: Flowlet and Field Tokenization.

To overcome this issue, we propose **Flowlet and Field Tokenization** as shown in Figure 1. Flowlet is a fine-grained flow segment and coherent behavior unit that groups packets from the same logical interaction (e.g., an HTTP request-response or a media stream). Flowlet also emphasizes Temporal Correlation, ensuring that packets transmitted within the same time frame are analyzed together. And Field Tokenization transforms each flowlet into discrete tokens based on protocol header fields, preserving field-level semantics for model input. Flowlet and Field Tokenization consists of 3 steps: **Flow Construction**, **Flowlet Generation** and **Field Tokenization**.

Flow Construction. Raw traffic is unordered and mixes multiple protocols, making pattern learning difficult. To impose semantic structure, we group packets by identical five-tuples and follow the relevant RFCs (Postel 1981a; Eddy 2022; Postel 1980, 1981b) to form flows.

Flowlet Generation. Consider a flow \mathbf{F} consisting of a sequence of n packets, denoted as $\mathbf{F} = \{pkt_1, pkt_2, \dots, pkt_n\}$. Each packet pkt_i has an arrival timestamp τ_i . The objective of Flowlet Generation is to segment this flow into multiple flowlets based on Inter-Arrival Time (IAT) between consecutive packets.

Let us define the IAT between consecutive packets as $t_i = \tau_i - \tau_{i-1}$ for $i \in 2, 3, \dots, n$. We introduce a dynamic threshold θ_i to determine flowlet boundaries, which is adaptively adjusted based on the historical IATs. Let W_i denote the IAT window up to the i -th packet. The threshold is calculated as:

$$\theta_i = \frac{1}{|W_i|} \sum_{t \in W_i} t$$

For each flowlet $\mathcal{F}_j = \{pkt_a, pkt_{a+1}, \dots, pkt_b\}$, the inter-arrival times within the flowlet satisfy:

$$t_i \leq \theta_{i-1}, \quad \forall i \in \{a+1, \dots, b\}.$$

If pkt_b is the last packet of flowlet \mathcal{F}_j , and pkt_{b+1} is the first packet of flowlet \mathcal{F}_{j+1} , then:

$$t_{b+1} > \theta_b.$$

The algorithm starts by constructing the first flowlet from the first packet and then processes each packet sequentially.

When $i > 3$ and the current IAT t_i exceeds the threshold θ_{i-1} , a new flowlet boundary is established. Otherwise, the packet is added to the current flowlet. The algorithm continuously updates the window W_i and adjusts the threshold accordingly to adapt to changing network conditions. More details about flowlet are provided in the Appendix B.

Field Tokenization. We transform Flowlets into tokens that can be input into the model. For each packet in the flowlet, we first extract the raw hexadecimal sequences (e.g. `0x45000034...`). Field tokenization is based on the length of protocol header fields, encoding the sequence into multiple hexadecimal tokens (e.g. `0x45000034...` into `4 5 00 0034...`). For fields longer than 2 bytes and payload, we split them into multiple 4-digit hexadecimal tokens to ensure uniformity and consistency in the model input format.

In this work, we use word-based tokenization (Mielke et al. 2021) rather than subword tokenization (Chung, Cho, and Bengio 2016; Sennrich, Haddow, and Birch 2016; Luong and Manning 2016), such as BPE (Sennrich, Haddow, and Birch 2016; Gage 1994) or WordPiece (Wu et al. 2016). The motivation behind this is that, we treat protocol header fields as the morpheme (smallest semantic units) in traffic, similar to individual characters in Chinese. In such languages, each character is a complete and indivisible unit of meaning. Likewise, each protocol field inherently carries distinct and atomic semantics, and should not be further split or processed using subword tokenization methods.

The maximum vocabulary size, denoted as $|V|$, is 65,812. This includes all possible tokens: 1-hex tokens (16 values), 2-hex tokens (256 values), 4-hex tokens (65,536 values), and five special tokens ([CLS], [SEP], [PAD], [MASK], [UNK]).

Model Architecture

FlowletFormer adopts the BERT-base model architecture (Devlin et al. 2019), which consists of two modules: an Embedding Module and a Transformer Encoder Module, as illustrated in Figure 2.

Embedding Module. Most existing pre-training models for traffic classification directly adopt the embedding module design for NLP, including token, position, and segment embedding. However, directly using these embeddings may

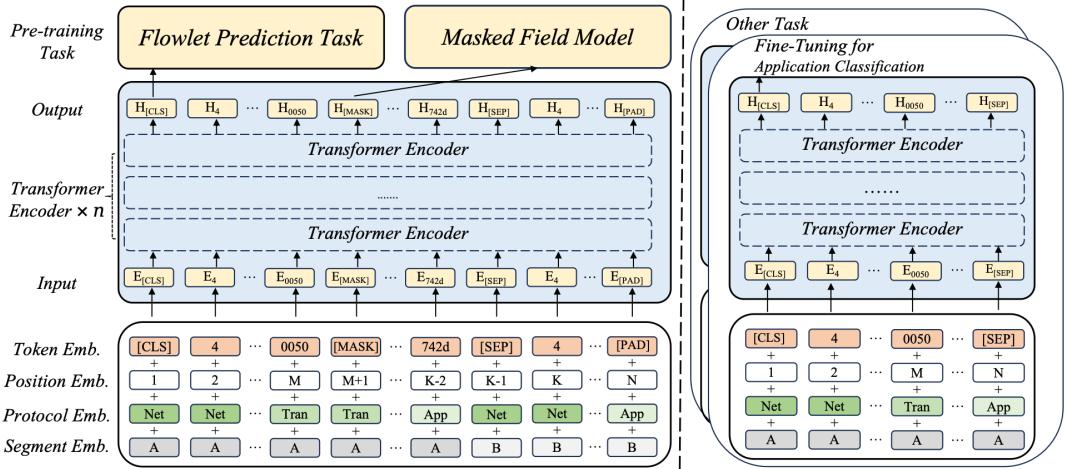


Figure 2: The flowchart of the FlowletFormer.

overlook the unique characteristics of traffic. Unlike natural language, traffic has a layered protocol structure, different forms of alignment and distribution.

Thus, we introduce a **Protocol Stack Alignment-Based Embedding Layer** into the existing embedding module. This embedding layer is specifically designed for traffic data and explicitly encodes the protocol layer associated with each token. Specifically, this embedding distinguishes between the network layer, transport layer, and application layer based on the TCP/IP model (Kurose and Ross 2001), and assigns each token an embedding corresponding to its protocol layer.

This design captures the semantic differences between different protocol layers. The model can not only process tokens based on their positions and sequential order, but also understand their functional roles within the protocol layer. This enables a hierarchical representation of traffic.

Finally, the embedding dimension is set to $D = 768$ and the input are calculated by the sum of each embedding layer:

$$\mathbf{E}_{\text{input}} = \mathbf{E}_{\text{token}} + \mathbf{E}_{\text{position}} + \mathbf{E}_{\text{segment}} + \mathbf{E}_{\text{protocol}} \quad (1)$$

Transformer Encoder Module. FlowletFormer is built on the BERT-Base architecture and contains 12 transformer encoder layers (Vaswani 2017), each with 12 multi-head self-attention heads and a position-wise feedforward network. Residual connections and layer normalization throughout the model ensure stable training and faster convergence. The total number of parameters is approximately 110 million. The number of input tokens is 512, and the dimension of each input token is 768.

Pre-training Method

We introduce the two novel pretraining tasks separately, followed by the overall loss function.

Masked Field Model. The masked modeling task randomly masks tokens and predicts the masked. Previous studies typically use this task to learn context and dependencies. However, in network traffic, the context and dependen-

Protocol	Key Fields
IP	Version, Total Length, Protocol, IP Address
TCP	Port Number, Sequence Number, Flag, Acknowledgment Number, Window Size
UDP	Port Number
ICMP	Type, Code

Table 1: Key fields in common protocol.

cies carried by different tokens vary in importance. Random masking may not fully capture the structural characteristics.

Thus, we introduce the Masked Field Model. During pre-training, 15% of the tokens in the input sequence are masked. **Half of these masked tokens are randomly selected from the key field tokens** mentioned in Table 8, while the other half are randomly selected from the remaining tokens. For the masked tokens, we replace them with the token [MASK], a random token, or leave them unchanged with probabilities of 80 %, 10 %, and 10 %, respectively.

For masked tokens, FlowletFormer must predict the token based on the context during pre-training. The loss function used is the cross-entropy loss, as shown in Equation 2.

$$\mathcal{L}_{\text{MFM}} = - \sum_{i \in \mathcal{M}_{\text{field}} \cup \mathcal{M}_{\text{random}}} m_i \log(\hat{m}_i) \quad (2)$$

Flowlet Prediction Task. Flowlet is generated based on the IAT between packets, which makes Flowlet more aligned with real network interactions, providing a better representation of network behavior and traffic patterns. For example, in a file download activity, a flow may represent the entire process of downloading the file, while **each Flowlet reflects specific behavior phases within the network interaction**, such as the request phase, download phase, and disconnection phase.

To better capture the diverse patterns in traffic, we introduce the Flowlet Prediction Task to predict the relationships

Dataset	ISCX-VPN(Service)				ISCX-Tor2016				ISCX-VPN(APP)				CSTNET-TLS			
	AC	PR	RC	F1	AC	PR	RC	F1	AC	PR	RC	F1	AC	PR	RC	F1
AppScanner	0.8681	0.8710	0.8435	0.8546	0.9075	0.7728	0.8033	0.7848	0.7945	0.6950	0.6975	0.6874	0.7441	0.7232	0.6963	0.7023
BIND	0.8345	0.7769	0.7714	0.7699	0.9010	0.8582	0.8354	0.8439	0.6951	0.6266	0.5415	0.5609	0.4710	0.4315	0.4226	0.4189
CUMUL	0.7099	0.6959	0.6893	0.6884	0.7725	0.6463	0.6443	0.6401	0.5480	0.4839	0.4615	0.4554	0.5921	0.5528	0.5604	0.5493
DF	0.5018	0.4664	0.4773	0.3934	0.7401	0.5918	0.5611	0.5492	0.4935	0.2449	0.2592	0.2289	0.5729	0.5398	0.5144	0.4933
FSNet	0.9087	0.9051	0.9054	0.9051	0.6967	0.6159	0.6061	0.6028	0.6316	0.4899	0.4833	0.4677	0.7814	0.7670	0.7316	0.7311
GraphDApp	0.7500	0.7311	0.7678	0.7429	0.7949	0.6391	0.6410	0.6383	0.5703	0.5108	0.4872	0.4853	0.7281	0.6964	0.6909	0.6890
Beauty	0.6416	0.5769	0.5842	0.5387	0.3746	0.2691	0.2767	0.2251	0.6169	0.3333	0.3139	0.2964	0.2944	0.3219	0.2513	0.2324
ET-BERT	0.8467	0.8496	0.8651	0.8393	0.8123	0.7249	0.8276	0.7453	0.7964	0.7332	0.7013	0.7066	0.7993	0.7832	0.7689	0.7700
YaTC	0.9010	0.8877	0.8800	0.8821	0.9175	0.7725	0.7333	0.7405	0.8214	0.7443	0.7265	0.7254	0.8391	0.8364	0.8101	0.8140
TrafficFormer	0.8533	0.8445	0.8348	0.8279	0.8669	0.7545	0.7460	0.7472	0.7751	0.7488	0.6846	0.6962	0.7982	0.7883	0.7736	0.7704
FlowletFormer	0.9400	0.9471	0.9277	0.9364	0.9215	0.9263	0.9043	0.9116	0.8480	0.8153	0.7641	0.7712	0.8605	0.8578	0.8445	0.8473

Table 2: Comparison results on ISCXPVN2016, ISCXTor2016, and CSTNET-TLS 1.3 datasets.

between Flowlets. During pre-training, we sample a pair of flowlets (\mathcal{F}_A , \mathcal{F}_B) and form the training instance. The pair is then drawn uniformly from three scenarios: \mathcal{F}_B is either the immediate successor of \mathcal{F}_A in the same flow (Ordered), the immediate predecessor (Swapped), or from a different flow. This design forces the model to learn intra-flow continuity, reverse-order dynamics, and separation of unrelated flowlets.

Unlike tasks based on individual packet or burst (Lin et al. 2022; Zhou et al. 2025), this task shifts the focus from individual packets to the relationships between behaviorally coherent Flowlets. Its goal is to capture the temporal and behavioral patterns of network traffic beyond the low-level semantics of individual packets.

Finally, the flowlet prediction task utilizes cross-entropy as the loss function, as shown in Equation 3.

$$\mathcal{L}_{\text{FPT}} = - \sum_{i=1}^N y_i \log(\hat{y}_i) \quad (3)$$

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

$$\mathcal{L} = \mathcal{L}_{\text{MFM}} + \mathcal{L}_{\text{FPT}} \quad (4)$$

Fine-tuning Method

FlowletFormer acquires generalizable knowledge during pre-training, capturing diverse patterns in traffic rather than being restricted to a specific task. This broad understanding enhances its transferability to various downstream applications. During fine-tuning, the pre-training knowledge is adapted to downstream traffic classification tasks.

Experiment

Experiment Setup

Pre-training Dataset. In this work, approximately 30GB of unlabeled raw traffic data is used for pre-training. The dataset was sourced from three main repositories: ISCXPVN2016 (NonVPN) (Draper-Gil et al. 2016), CIC-IDS2017 (Monday) (Sharafaldin, Lashkari, and Ghorbani 2018), and the WIDE backbone dataset (January 1, 2024) (Cho, Mitsuya, and Kato 2000). These datasets encompass a

variety of network application scenarios and protocols, such as web browsing with HTTP, file downloads with FTP, email with SMTP, and video streaming with QUIC.

During pre-training dataset construction, we extract 64 consecutive bytes from the beginning of the Network Layer of each packet as the model input, in order to cover key information from the IP layer and above. Furthermore, **no randomization was applied to the pre-training dataset**.

Fine-tuning Dataset. We employ 8 datasets for fine-tuning, corresponding to 7 different downstream tasks, including Service Type Identification (ISCX-VPN (Service) (Draper-Gil et al. 2016) and ISCXTor2016 (Lashkari et al. 2017)), Application Classification (ISCX-VPN (App) (Draper-Gil et al. 2016)), Website Fingerprinting (CSTNET-TLS (Lin et al. 2022)), Browser Classification (Browser (Liu et al. 2019a)), Malware Classification (USTC-TFC (Wang et al. 2017)), Malicious Traffic Classification (CIC-IDS2017 (Sharafaldin, Lashkari, and Ghorbani 2018)), and IoT Classification (CIC-IoT2022 (Dadkhah et al. 2022)).

During fine-tuning dataset construction, we select the first five packets of each flow and extract 64 bytes starting from the Network Layer of each packet. To preserve privacy and mitigate potential biases, **we further anonymize the packets by applying IP Address&Port randomization and TCP timestamp adjustments**. The pretraining and fine-tuning datasets are strictly separated to avoid data leakage.

Evaluation Metrics. We adopt classification accuracy (AC), precision (PR), recall (RC), and F1 score as the evaluation metrics for the experiments. More Experiment Setup details are shown in Appendix D.

Comparison with State-of-the-Art Methods

We compare FlowletFormer with various baselines and state-of-the-art methods. AppScanner (Taylor et al. 2016), BIND (Al-Naami et al. 2016), and CUMUL (Panchenko et al. 2016) are based on ML models. DF (Sirinam et al. 2018), FSNet (Liu et al. 2019a), GraphDapp (Shen et al. 2021) and Beauty (Schuster, Shmatikov, and Tromer 2017) use DL models. ET-BERT (Lin et al. 2022), YaTC (Zhao et al. 2023) and TrafficFormer (Zhou et al. 2025) are pre-training methods. All pretraining methods are trained on the same pre-training dataset and fine-tuning dataset. The ML and DL models are trained on fine-tuning dataset.

Dataset	Browser				USTC-TFC				CIC-IDS2017				CIC-IoT2022			
	AC	PR	RC	F1												
AppScanner	0.5825	0.5836	0.6023	0.5733	0.8585	0.9108	0.9034	0.8976	0.8712	0.8820	0.8697	0.8630	0.8591	0.8858	0.7996	0.8288
BIND	0.5842	0.5875	0.5704	0.5738	0.7945	0.7811	0.7061	0.7115	0.9114	0.9252	0.8652	0.8788	0.7349	0.6754	0.6387	0.6435
CUMUL	0.5253	0.5226	0.5202	0.5207	0.7173	0.5063	0.5812	0.5183	0.8478	0.6867	0.7160	0.6951	0.7019	0.6746	0.7029	0.6687
DF	0.2450	0.1966	0.2454	0.1627	0.6452	0.4019	0.3685	0.3059	0.6672	0.6756	0.6348	0.5940	0.2746	0.2140	0.1870	0.1647
FSNet	0.6405	0.6424	0.6491	0.6410	0.7558	0.8167	0.8407	0.8042	0.8436	0.8198	0.8106	0.8144	0.8077	0.8250	0.8333	0.7804
GraphDApp	0.4844	0.5005	0.4969	0.4912	0.8750	0.8446	0.8501	0.8249	0.8750	0.8989	0.8515	0.8647	0.7370	0.6721	0.7006	0.6767
Beauty	0.2250	0.1393	0.2364	0.1040	0.6682	0.4448	0.4369	0.3796	0.6641	0.7720	0.6482	0.6567	0.1356	0.0349	0.0764	0.0296
ET-BERT	0.4700	0.4861	0.4700	0.3439	0.9663	0.9711	0.9663	0.9666	0.8950	0.9000	0.8950	0.8911	0.8603	0.8297	0.8255	0.8244
YatC	0.5276	0.5348	0.5278	0.5154	0.9712	0.9732	0.9712	0.9707	0.9083	0.9332	0.9083	0.8959	0.8448	0.8656	0.8074	0.8048
TrafficFormer	0.4750	0.7366	0.4750	0.3320	0.9750	0.9789	0.9750	0.9746	0.8783	0.8801	0.8783	0.8785	0.8725	0.8487	0.8343	0.8288
FlowletFormer	0.7050	0.7742	0.7050	0.6684	0.9650	0.9689	0.9650	0.9648	0.9183	0.9475	0.9183	0.9079	0.9109	0.8905	0.8866	0.8859

Table 3: Comparison results on Browser, USTC-TFC, CIC-IDS2017, and CIC-IoT2022 datasets.

As shown in Table 2 and 3, FlowletFormer outperforms all methods on 7 datasets. Especially in the Service Type Identification (ISCX-VPN Service, Tor) task, FlowletFormer attains an F1-score of 0.9364 and 0.9116, outperforming the second-best methods, FSNet and ET-BERT, by 3.1% and 16.6%, respectively. Even in the Malware Classification Task, FlowletFormer is only 1% lower than the best performing method (TrafficFormer). In fact, this trivial difference largely varies due to random data splits; we show in Appendix E that FlowletFormer outperforms TrafficFormer in 3 of the other 5 splits using different random seeds on this dataset. These results demonstrate that FlowletFormer, as a pre-training model with a more specific design for traffic, can flexibly and effectively adapt to various traffic classification tasks in diverse network environments, indicating its promise in improving network management and security.

Ablation Study

To evaluate the contribution of different components in FlowletFormer, we conduct an ablation study on 2 datasets. Specifically, we systematically remove key components, including FlowLet, the Masked Field Model, the Flowlet Prediction Task, the Protocol Stack Alignment-Based Embedding Layer, and the Pre-training stage. The results are presented in Figure 3. First, Flowlet significantly improves recall and overall F1 score by capturing fine-grained Flowlet dependencies, while the Masked Field Model helps the model learn structured contextual representations, leading to better generalization. The Flowlet Prediction Task contributes to flowlet-level continuity modeling, which is essential for robust classification. The Protocol Stack Alignment-Based Embedding Layer provides marginal improvements, indicating its role in refining hierarchical representations. Most notably, the Pre-training is the most important, as removing it leads to a drastic performance drop, highlighting its importance in learning general traffic representations. The details are provided in Appendix E.

Few-shot Analysis

To further assess the effectiveness and robustness of FlowletFormer under few-shot conditions, we conduct experiments with varying data proportions on 2 datasets. Specifically,

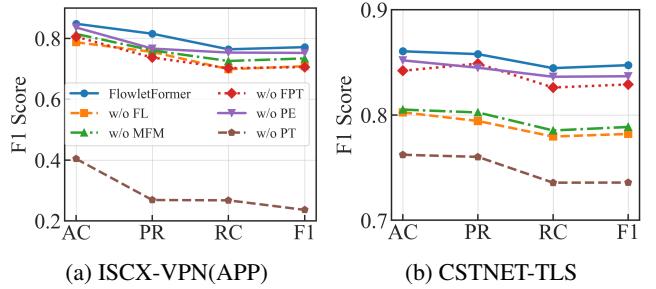


Figure 3: **Ablation Study of key components in FlowletFormer.** The abbreviations are explained as follows: FL: Flowlet and Field Tokenization, MFM: Masked Field Model, FPT: Flowlet Prediction Task, PE: Protocol Stack Alignment-Based Embedding Layer, and PT: Pre-Training.

we use the full dataset as the reference and randomly sample 40%, 20%, and 10% of the available data for few-shot training. Our few-shot evaluation on ISCX-VPN-App reveals FlowletFormer’s superior data efficiency, maintaining F1 scores of 0.8009 (40% data), 0.6224 (20%), and 0.5813 (10%) – surpassing state-of-the-art baselines by 12.7-46.1% margins. Notably, while supervised methods (e.g., BIND/DF/FSNet) exhibit catastrophic performance under data scarcity (DF declines 46.1% at 10% data), our pre-training framework maintains performance through the traffic representation model, as evidenced in Figure 4. More results can be found in Appendix E.

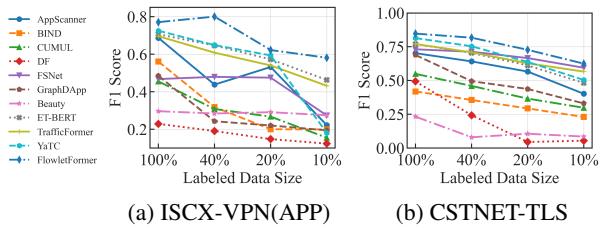


Figure 4: **Few-shot Analysis.**

Task	Flow Direction Inference			Transport Protocols Recognition			Sequence Awareness			Connection control Judgement		
Dataset	VPN	IDS	TFC	VPN	IDS	TFC	VPN	IDS	TFC	VPN	IDS	TFC
ET-BERT	0.4366	0.7096	0.7412	0.9681	0.9767	0.9981	0.4165	0.6937	0.6203	0.9041	0.9975	0.9985
TrafficFormer	0.0164	0.1059	0.1128	0.6753	0.9067	0.8912	0.3659	0.5261	0.3652	0.3904	0.9983	0.9978
FlowletFormer	0.9313	0.9647	0.9196	1.0000	1.0000	1.0000	0.6987	0.7806	0.7579	0.9338	1.0000	1.0000

Table 4: The performance comparison with other pre-training models on field understanding tasks.

Field Understanding Task

We introduce multiple **Field Understanding Tasks** to assess whether the pre-trained model¹ comprehends general traffic patterns. These tasks require the model to predict key header fields within a packet in a given flow. Specifically, we evaluate the comprehension of the model in four tasks: the **Flow Direction Inference** task masks the source/destination IP and Ports, assessing the model’s ability to infer packet direction between entities based on contextual clues without direct address information; the **Transport Protocol Recognition** task focuses on masking the protocol field in the IP header, testing the model’s ability to identify the transport layer protocol (e.g., TCP, UDP, ICMP); the **Sequence Awareness** task masks the sequence number and acknowledgment number within the TCP header, challenging the model to infer packet order and flow continuity; the **Connection Control Judgment** task masks the flag fields in the TCP header, which denote the state of the connection, and evaluates the model’s ability to infer control signals like session establishment or termination.

These tasks evaluate the model’s ability to infer direction, protocol, sequence, and control, with performance measured in three datasets: ISCX VPN, CICIDS2017, and USTC-TFC. As shown in Table 4, FlowletFormer outperforms two models in all tasks. The model’s ability to effectively infer Flow Direction, Transport Protocol, Sequence Awareness, and Connection Control across diverse datasets demonstrates its strong capacity for understanding the complex behavior of network traffic.

Word Analogies Similarity Analysis

In NLP, word analogy tasks assess a model’s ability to capture semantic relationships between words. Through word analogy similarity analysis, we can validate whether a model has deeply understood the semantic relationships between words. Similarly, the port number analogy analysis can be used to evaluate the pre-trained model¹, assessing its understanding of the functional and semantic relationships between network services. This capability reflects the model’s deep understanding of traffic patterns acquired during pre-training, without any downstream fine-tuning.

We apply cosine similarities between the embeddings of port numbers produced by the pre-trained model to examine the relationships among common HTTP-related ports (e.g., 80, 8080, 8000). Comparing ET-BERT with FlowletFormer (Table 5), we find that ET-BERT struggles to model port similarities, while FlowletFormer effectively captures these relationships, enhancing traffic classification performance. Appendix E provides more clarification.

Port	80&8080		80&8000		8080&8000	
	Word	Input	Word	Input	Word	Input
ET-BERT	-0.0768	0.1094	-0.0685	0.1331	0.0740	0.2438
FlowletFormer	0.0582	0.4019	0.0369	0.3993	0.0400	0.4289

Table 5: Port Number Analogy Cosine Similarity about Word Embedding and Input Embedding.

Computational Cost and Complexity

We analyze the time complexity of our method. Since our model is built upon the BERT-Base architecture, it shares the same pre-training time complexity as the two word-like methods, ET-BERT and TrafficFormer. Specifically, the complexity is:

$$\mathcal{O}(N \times B \times L \times (S^2 \cdot H + S \cdot H^2)) \quad (5)$$

where N is the number of training steps, B is the batch size, L is the number of Transformer layers, S is the input sequence length, and H is the hidden size. We also measure the end-to-end runtimes of FlowletFormer during different phases of the train. Table 6 summarizes these results.

Phase	GPUs	Time	Unit/Granularity	GPU Memory (GB)
Pre-training	6	42 h	75.67 s / 100 steps	28
Fine-tuning	1	1,153 s	57.65 s / epoch	17
Inference	1	–	150.04 samples/sec	–

Table 6: Computational Efficiency Across Different Phases about FlowletFormer. More detail are shown in Appendix E.

Conclusion

In this paper, we propose FlowletFormer, a BERT-based pre-training model designed for network traffic analysis. By introducing a Coherent Behavior-Aware Traffic Representation Model, a Protocol Stack Alignment-Based Embedding Layer, and Field-Specific and Context-Aware Pretraining Tasks, FlowletFormer effectively captures behavioral patterns, hierarchical protocol semantics, and inter-packet contextual relationships among traffic data. The experimental results demonstrate its superiority over existing methods in traffic classification.

FlowletFormer improves network traffic classification, but challenges remain. Future work includes adapting to evolving traffic patterns, enhancing robustness against adversarial attacks, incorporating multi-modal data, and optimizing computational efficiency for real-time deployment. Addressing these issues will strengthen its role in network security and traffic classification.

References

- Al-Naami, K.; Chandra, S.; Mustafa, A.; Khan, L.; Lin, Z.; Hamlen, K. W.; and Thuraisingham, B. 2016. Adaptive encrypted traffic fingerprinting with bi-directional dependence. In Schwab, S.; Robertson, W. K.; and Balzarotti, D., eds., *Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC 2016, Los Angeles, CA, USA, December 5-9, 2016*, 177–188. ACM.
- Aouedi, O.; Piamrat, K.; Hamma, S.; and Kuranage, M. P. J. 2022. Network traffic analysis using machine learning: an unsupervised approach to understand and slice your network. *Ann. des Télécommunications*, 77(5-6): 297–309.
- Cho, K.; Mitsuya, K.; and Kato, A. 2000. Traffic Data Repository at the WIDE Project. In *Proceedings of the Freenix Track: 2000 USENIX Annual Technical Conference, June 18-23, 2000, San Diego, CA, USA*, 263–270. USENIX.
- Chung, J.; Cho, K.; and Bengio, Y. 2016. A Character-level Decoder without Explicit Segmentation for Neural Machine Translation. *CoRR*, abs/1603.06147.
- Dadkhah, S.; Mahdikhani, H.; Danso, P. K.; Zohourian, A.; Truong, K. A.; and Ghorbani, A. A. 2022. Towards the Development of a Realistic Multidimensional IoT Profiling Dataset. In *19th Annual International Conference on Privacy, Security & Trust, PST 2022, Fredericton, NB, Canada, August 22-24, 2022*, 1–11. IEEE.
- Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Burstein, J.; Doran, C.; and Solorio, T., eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 4171–4186. Association for Computational Linguistics.
- Draper-Gil, G.; Lashkari, A. H.; Mamun, M. S. I.; and Ghorbani, A. A. 2016. Characterization of Encrypted and VPN Traffic using Time-related Features. In Camp, O.; Furnell, S.; and Mori, P., eds., *Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP 2016, Rome, Italy, February 19-21, 2016*, 407–414. SciTePress.
- Eddy, W. 2022. Rfc 9293: Transmission control protocol (tcp).
- Gage, P. 1994. A new algorithm for data compression. *C Users J.*, 12(2): 23–38.
- Guterman, C.; Guo, K.; Arora, S.; Wang, X.; Wu, L.; Katz-Bassett, E.; and Zussman, G. 2019. Requet: real-time QoE detection for encrypted YouTube traffic. In Zink, M.; Toni, L.; and Begen, A. C., eds., *Proceedings of the 10th ACM Multimedia Systems Conference, MMSys 2019, Amherst, MA, USA, June 18-21, 2019*, 48–59. ACM.
- He, H. Y.; Yang, Z. G.; and Chen, X. N. 2020. PERT: Payload Encoding Representation from Transformer for Encrypted Traffic Classification. In *2020 ITU Kaleidoscope: Industry-Driven Digital Transformation, Kaleidoscope, Ha Noi, Vietnam, December 7-11, 2020*, 1–8. IEEE.
- Hu, X.; Gao, W.; Cheng, G.; Li, R.; Zhou, Y.; and Wu, H. 2023. Toward Early and Accurate Network Intrusion Detection Using Graph Embedding. *IEEE Trans. Inf. Forensics Secur.*, 18: 5817–5831.
- Kurose, J. F.; and Ross, K. W. 2001. *Computer networking - a top-down approach featuring the internet*. Addison-Wesley-Longman. ISBN 978-0-201-47711-5.
- Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- Lashkari, A. H.; Draper-Gil, G.; Mamun, M. S. I.; and Ghorbani, A. A. 2017. Characterization of Tor Traffic using Time based Features. In Mori, P.; Furnell, S.; and Camp, O., eds., *Proceedings of the 3rd International Conference on Information Systems Security and Privacy, ICISSP 2017, Porto, Portugal, February 19-21, 2017*, 253–262. SciTePress.
- Lin, X.; Xiong, G.; Gou, G.; Li, Z.; Shi, J.; and Yu, J. 2022. ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification. In Laforest, F.; Troncy, R.; Simperl, E.; Agarwal, D.; Gionis, A.; Herman, I.; and Médini, L., eds., *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, 633–642. ACM.
- Liu, C.; He, L.; Xiong, G.; Cao, Z.; and Li, Z. 2019a. FS-Net: A Flow Sequence Network For Encrypted Traffic Classification. In *2019 IEEE Conference on Computer Communications, INFOCOM 2019, Paris, France, April 29 - May 2, 2019*, 1171–1179. IEEE.
- Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019b. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, abs/1907.11692.
- Lotfollahi, M.; Siavoshani, M. J.; Zade, R. S. H.; and Saberian, M. 2020. Deep packet: a novel approach for encrypted traffic classification using deep learning. *Soft Comput.*, 24(3): 1999–2012.
- Luong, M.; and Manning, C. D. 2016. Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models. *CoRR*, abs/1604.00788.
- Mao, H.; Schwarzkopf, M.; Venkatakrishnan, S. B.; Meng, Z.; and Alizadeh, M. 2019. Learning scheduling algorithms for data processing clusters. In Wu, J.; and Hall, W., eds., *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019*, 270–288. ACM.
- Mielke, S. J.; Alyafeai, Z.; Salesky, E.; Raffel, C.; Dey, M.; Gallé, M.; Raja, A.; Si, C.; Lee, W. Y.; Sagot, B.; et al. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *arXiv preprint arXiv:2112.10508*.
- Miller, B.; Huang, L.; Joseph, A. D.; and Tygar, J. D. 2014. I Know Why You Went to the Clinic: Risks and Realization of HTTPS Traffic Analysis. In Cristofaro, E. D.; and Murdoch,

- S. J., eds., *Privacy Enhancing Technologies - 14th International Symposium, PETS 2014, Amsterdam, The Netherlands, July 16-18, 2014. Proceedings*, volume 8555 of *Lecture Notes in Computer Science*, 143–163. Springer.
- Panchenko, A.; Lanze, F.; Pennekamp, J.; Engel, T.; Zinnen, A.; Henze, M.; and Wehrle, K. 2016. Website Fingerprinting at Internet Scale. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society.
- Papadogiannaki, E.; and Ioannidis, S. 2022. A Survey on Encrypted Network Traffic Analysis Applications, Techniques, and Countermeasures. *ACM Comput. Surv.*, 54(6): 123:1–123:35.
- Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.
- Postel, J. 1980. Rfc 0768: User datagram protocol.
- Postel, J. 1981a. Internet protocol. Technical report.
- Postel, J. 1981b. Rfc 792: Internet Control Message Protocol darpa internet program protocol specification.
- Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I.; et al. 2018. Improving language understanding by generative pre-training.
- Rezaei, S.; and Liu, X. 2019. Deep Learning for Encrypted Traffic Classification: An Overview. *IEEE Commun. Mag.*, 57(5): 76–81.
- Roesch, M. 1999. Snort: Lightweight Intrusion Detection for Networks. In Parter, D. W., ed., *Proceedings of the 13th Conference on Systems Administration (LISA-99), Seattle, WA, USA, November 7-12, 1999*, 229–238. USENIX.
- Schuster, R.; Shmatikov, V.; and Tromer, E. 2017. Beauty and the Burst: Remote Identification of Encrypted Video Streams. In Kirda, E.; and Ristenpart, T., eds., *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, 1357–1374. USENIX Association.
- Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.
- Sharafaldin, I.; Lashkari, A. H.; and Ghorbani, A. A. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Mori, P.; Furnell, S.; and Camp, O., eds., *Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP 2018, Funchal, Madeira - Portugal, January 22-24, 2018*, 108–116. SciTePress.
- Shen, M.; Liu, Y.; Zhu, L.; Xu, K.; Du, X.; and Guizani, N. 2020. Optimizing Feature Selection for Efficient Encrypted Traffic Classification: A Systematic Approach. *IEEE Netw.*, 34(4): 20–27.
- Shen, M.; Zhang, J.; Zhu, L.; Xu, K.; and Du, X. 2021. Accurate Decentralized Application Identification via Encrypted Traffic Analysis Using Graph Neural Networks. *IEEE Trans. Inf. Forensics Secur.*, 16: 2367–2380.
- Sirinam, P.; Imani, M.; Juarez, M.; and Wright, M. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In Lie, D.; Mannan, M.; Backes, M.; and Wang, X., eds., *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, 1928–1943. ACM.
- Sommer, R.; and Paxson, V. 2010. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *31st IEEE Symposium on Security and Privacy, SP 2010, 16-19 May 2010, Berleley/Oakland, California, USA*, 305–316. IEEE Computer Society.
- Tang, R.; Yang, Z.; Li, Z.; Meng, W.; Wang, H.; Li, Q.; Sun, Y.; Pei, D.; Wei, T.; Xu, Y.; and Liu, Y. 2020. ZeroWall: Detecting Zero-Day Web Attacks through Encoder-Decoder Recurrent Neural Networks. In *39th IEEE Conference on Computer Communications, INFOCOM 2020, Toronto, ON, Canada, July 6-9, 2020*, 2479–2488. IEEE.
- Taylor, V. F.; Spolaor, R.; Conti, M.; and Martinovic, I. 2016. AppScanner: Automatic Fingerprinting of Smartphone Apps from Encrypted Network Traffic. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, 439–454. IEEE.
- van Ede, T.; Bortolameotti, R.; Continella, A.; Ren, J.; Dubois, D. J.; Lindorfer, M.; Choffnes, D. R.; van Steen, M.; and Peter, A. 2020. FlowPrint: Semi-Supervised Mobile-App Fingerprinting on Encrypted Network Traffic. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society.
- Vaswani, A. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.
- Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; and Sheng, Y. 2017. Malware traffic classification using convolutional neural network for representation learning. In *2017 International Conference on Information Networking, ICOIN 2017, Da Nang, Vietnam, January 11-13, 2017*, 712–717. IEEE.
- Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; Klingner, J.; Shah, A.; Johnson, M.; Liu, X.; Łukasz Kaiser; Gouws, S.; Kato, Y.; Kudo, T.; Kazawa, H.; Stevens, K.; Kurian, G.; Patil, N.; Wang, W.; Young, C.; Smith, J.; Riesa, J.; Rudnick, A.; Vinyals, O.; Corrado, G.; Hughes, M.; and Dean, J. 2016. Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv:1609.08144*.
- Zhang, J.; Li, F.; Ye, F.; and Wu, H. 2020. Autonomous Unknown-Application Filtering and Labeling for DL-based Traffic Classifier Update. In *39th IEEE Conference on Computer Communications, INFOCOM 2020, Toronto, ON, Canada, July 6-9, 2020*, 397–405. IEEE.
- Zhao, R.; Zhan, M.; Deng, X.; Wang, Y.; Wang, Y.; Gui, G.; and Xue, Z. 2023. Yet Another Traffic Classifier: A Masked Autoencoder Based Traffic Transformer with Multi-Level Flow Representation. In Williams, B.; Chen, Y.; and

Neville, J., eds., *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, 5420–5427. AAAI Press.

Zhao, Z.; Chen, H.; Zhang, J.; Zhao, X.; Liu, T.; Lu, W.; Chen, X.; Deng, H.; Ju, Q.; and Du, X. 2019. UER: An Open-Source Toolkit for Pre-training Models. 241–246.

Zhou, G.; Guo, X.; Liu, Z.; Li, T.; Li, Q.; and Xu, K. 2025. TrafficFormer: An Efficient Pre-trained Model for Traffic Data . In *2025 IEEE Symposium on Security and Privacy (SP)*, 102–102. IEEE Computer Society.

Zuev, D.; and Moore, A. W. 2005. Traffic Classification Using a Statistical Approach. In Dovrolis, C., ed., *Passive and Active Network Measurement, 6th International Workshop, PAM 2005, Boston, MA, USA, March 31 - April 1, 2005, Proceedings*.

Appendix

Preliminary Analysis

We conduct three in-depth analyses to examine the limitations of existing tokenization and segmentation strategies for traffic data representation:

First, we investigate the applicability of sub-word tokenization techniques, which are widely adopted in NLP to reduce vocabulary size. Methods such as ET-BERT and TrafficFormer directly employ byte-pair encoding (BPE) to tokenize traffic data. To evaluate its effectiveness in this domain, we apply sub-word tokenization to the CICIDS2017-Monday dataset. Statistical results show that **less than 1% of the tokens are segmented into subwords** (e.g., `1df8` into `1df` and `#8`), indicating that most tokens remain unsplit. This suggests that such techniques may offer limited granularity and expressiveness when applied to traffic data.

Second, we identify a critical limitation in how existing methods often overlook the hierarchical structure of packet data. Unlike natural language, network packets are composed of protocol layers with distinct semantics (e.g., IP, TCP, application layer). Flattening these layered structures into a linear token sequence may result in a loss of structural information, hindering the model’s capacity to learn meaningful representations.

Third, we analyze burst segmentation in the CICIDS2017-Monday dataset. Bursts are commonly used to reflect traffic behaviors in application layer. We compute the cumulative distribution function (CDF) of the number of packets per burst and find that **approximately 65% of bursts consist of only a single packet**. This suggests that the dataset contains a high proportion of extremely short bursts, which limits the temporal context available for modeling. Moreover, since existing pretraining tasks (e.g., Same-origin Burst Prediction) often rely on splitting bursts into sub-bursts or sub-packets, the semantic connections between these sub-units are likely to be weak. Consequently, the model may struggle to learn robust traffic patterns from such limited contexts.

More Details of Our Method

Flow Construction To construct semantically meaningful flows from raw packet data, we apply protocol-specific rules according to standard practices outlined in RFCs and previous works. The flow construction process is based on the five-tuple: `srcIP`, `dstIP`, `srcPort`, `dstPort`, `protocol`, with additional considerations depending on the transport layer protocol.

We apply protocol-specific rules based on both packet semantics and timeout heuristics. As shown in Table 7, different protocols adopt distinct termination and reinitialization criteria. For instance, TCP flows are explicitly closed by a four-way handshake or reset flag, while UDP and ICMP rely on timeout-based or field-change-based segmentation. These rules help segment raw traffic into coherent flow units for downstream analysis.

Flowlet Generation After flow construction, we perform the Flowlet Generation. We also describe it in Algorithm 1

The Flowlet Generation Algorithm dynamically partitions a flow into flowlets based on inter-packet arrival time. It operates as follows:

- **Initialization:** For each network flow $F = \{\text{pkt}_1, \dots, \text{pkt}_n\}$ with timestamps $\{\tau_1, \dots, \tau_n\}$, we compute the average inter-arrival time of the first three packets, i.e., $\theta_3 = \frac{1}{2}[(\tau_2 - \tau_1) + (\tau_3 - \tau_2)]$. This value is used as the initial threshold θ for segmentation. If $n \leq 3$, the entire flow is treated as a single Flowlet.
- **Segmentation:** For each subsequent packet pkt_i ($i > 3$), we calculate the inter-arrival time $t_i = \tau_i - \tau_{i-1}$. If $t_i > \theta_{i-1}$, we create a segmentation: the previous packet pkt_{i-1} ends the current Flowlet \mathcal{F}_j , and pkt_i begins a new one \mathcal{F}_{j+1} . Otherwise, pkt_i is appended to the current \mathcal{F}_j .
- **Threshold Update:** After each decision, we update the threshold θ_i using all observed inter-arrival times up to index i , i.e., $\theta_i = \frac{1}{|W_i|} \sum_{t \in W_i} t$, where W_i is the window of past IATs. This allows the threshold to adapt dynamically to local flow patterns.

This adaptive thresholding approach allows the segmentation process to adjust to diverse traffic dynamics. For instance, traffic patterns such as HTTP request-response cycles or video streaming often exhibit short bursts followed by longer silent gaps. By capturing such timing structures, Flowlet segmentation enables the model to better align with the logical behavior units within network communication, thus enhancing the semantic granularity of traffic representation.

Algorithm 1: Flowlet Generation

```

1: Input: Flow  $F = \{\text{pkt}_1, \dots, \text{pkt}_n\}$  with arrival times-
   tamps  $\{\tau_1, \dots, \tau_n\}$ 
2: Output: Flowlets  $\{\mathcal{F}_1, \dots, \mathcal{F}_k\}$ 
3: Initialize:  $\mathcal{F} \leftarrow \{\text{pkt}_1\}$ ,  $W \leftarrow \emptyset$ ,  $\text{flowlets} \leftarrow \emptyset$ 
4: for  $i \leftarrow 2$  to  $n$  do
5:    $t_i \leftarrow \tau_i - \tau_{i-1}$ 
6:   if  $i > 3$  and  $t_i > \theta_{i-1}$  then
7:     Append  $\mathcal{F}$  to  $\text{flowlets}$ 
8:      $\mathcal{F} \leftarrow \{\text{pkt}_i\}$ 
9:   else
10:    Append  $\text{pkt}_i$  to  $\mathcal{F}$ 
11:   end if
12:   Append  $t_i$  to  $W$ 
13:    $\theta_i \leftarrow \frac{1}{|W|} \sum_{t \in W} t$ 
14: end for
15: Append remaining  $\mathcal{F}$  to  $\text{flowlets}$ 

```

Key Protocol Header Fields in Masked Field Model Table 8 lists the key fields commonly found in standard network protocols. These fields carry rich semantic and structural information that can be leveraged by traffic analysis models.

For example, fields such as IP addresses, port numbers, and protocol types provide fundamental information about the directionality and service type of a packet, helping mod-

Table 7: Protocol-specific Rules for Flow Construction.

Protocol	Flow Termination Condition	New Flow Trigger
TCP	Four-way Handshake (FIN + FIN + ACK)	New SYN + ACK Connection
	Connection Reset (RST packet)	Active Timeout Expiration
	Active Timeout (Flow duration exceeds 1800s)	
UDP	Inactive Timeout (Flow duration exceeds 15s)	Inactive Timeout Expiration
ICMP	Change in ICMP Type Change in ICMP Code	Any change in Type or Code
Others	Flow duration exceeds 1800 seconds	Timeout Expiration

els distinguish between client-server roles or application types.

Sequence Number and Acknowledgment Number in the TCP header reflect the transmission order and reliability mechanisms of the protocol, offering temporal cues to infer packet sequences and session continuity.

The Total Length field, which indicates the size of an entire packet, has been demonstrated to serve as an effective signature for encrypted traffic classification in prior studies.

Furthermore, TCP control flags (e.g., SYN, ACK, FIN, RST) encode connection state transitions (e.g., handshake, termination), enabling models to learn flow dynamics and session boundaries.

Similarly, ICMP’s Type and Code fields identify message semantics (e.g., echo request/reply, destination unreachable), while the minimal set of fields in UDP (primarily source and destination ports) still conveys important endpoint semantics.

Table 8: Key fields in common protocol.

Protocol	Key Fields
IP	Version, Total Length, Protocol, IP Address
TCP	Port Number, Sequence Number, Flag
	Acknowledgment Number, Window Size
UDP	Port Number
ICMP	Type, Code

More Details in Experiment Setup

More Details in Pre-training Dataset Construction We describe the data preprocessing pipeline used during the pre-training stage of FlowletFormer.

Flow Construction. We first parsed raw PCAP files to construct flows based on five-tuples and protocol-specific rules which ensure semantically coherent flow boundaries. Each flow was saved as an individual PCAP file for subsequent processing.

Flowlet Segmentation. To better reflect the temporal structure and traffic behavior from application layer, we further segmented each flow into multiple flowlets. Specifically,

we calculated inter-packet arrival times (IATs) and initiated a new flowlet whenever the IAT exceeded a threshold. This segmentation captures distinct behavioral units within each flow and enables the model to learn fine-grained communication patterns.

Tokenization. For each packet in a flowlet, we removed the Ethernet header and retained the first 64 bytes starting from the network layer. These bytes were tokenized using Field Tokenization, where individual fields in protocol headers (e.g., IP version, TTL, TCP flags) are identified and converted into semantically meaningful tokens. This tokenization approach preserves protocol semantics while producing a consistent and structured input format for the model.

Table 9 summarizes the pre-training datasets used in this work, including their sizes, number of flows, and supported protocols.

More Details in Fine-tuning Dataset Construction To ensure fair comparison and reproducibility, we describe the data preprocessing pipeline used during the fine-tuning stage of FlowletFormer.

Data Collection and Filtering. We collected raw PCAP files corresponding to the eight downstream tasks. Flows were constructed based on five-tuples (srcIP, dstIP, srcPort, dstPort, protocol), and each flow was saved as a separate PCAP file.

Flows were then organized by traffic category. To facilitate manageable storage and training, large files were split into smaller ones (approximately 1,000 packets each). Categories with fewer than 10 samples were discarded, and a maximum of 500 samples per class was retained to ensure balanced representation.

Data Anonymization and Randomization. To mitigate the risk of shortcut learning and reduce the model’s dependence on protocol-specific artifacts, we performed the following anonymization steps on each flow:

- Replaced all IP addresses with randomly generated addresses;
- Randomized source and destination ports while preserving client/server roles;
- Adjusted TCP timestamps by introducing a random base time, but preserving the relative inter-packet timing.

Tokenization. We selected the first five packets of each

Table 9: Overview of Pre-training Datasets.

Dataset	Size	Flow Number	Protocol
ISCX-VPN2016-NonVPN	10.4G	74,184	TLS1.2, SFTP, SSDP, SNMP, NTP, MDNS, HTTP, GQUIC...
CIC-IDS2017-Monday	11G	303,436	HTTP, HTTPS, FTP, SSH, email protocols...
WIDE-2024/1/1	9.6G	2,322,172	FTP, SSH, IPSec, HTTP, TLS1.2, TLS1.3, GRE, Email Protocol...

flow and converted their contents to input tokens. Each packet was tokenized by retaining the first 64 tokens.

Table 10 provides an overview of all downstream tasks used for fine-tuning FlowletFormer, including dataset names, number of flows, number of classes, and example labels.

More Details in Implementation In this experiment, we employ multi-GPU parallel in pre-training. A total of six GPUs are used for distributed training, with a batch size set to 16, resulting in an overall batch size of 96. The total number of training steps is 200,000, with model checkpoints saved every 10,000 steps. The Adam optimizer is chosen, with an initial learning rate of 2e-5 and a warm-up ratio of 0.1 to ensure stability during the initial stages of training.

To maintain consistency with pre-training, the fine-tuning data is processed in the same input format as the pre-training data. The packets in the flowlets are directly concatenated without [SEP] token for separation, meaning all tokens share the same segment identifiers. During the fine-tuning stage, we select the first five packets of each network flow as the model input and extract the first 64 tokens following the Ethernet header of each packet. The dataset is split into train/validation/test sets with an 8:1:1 ratio. The model was trained for up to 20 epochs on each dataset using the AdamW optimizer with a learning rate of 6e-5, with early stopping triggered if the F1 score did not improve for 4 consecutive epochs.

The proposed method is implemented using PyTorch 2.3.1 and UER and trained on a server with 8 NVIDIA Tesla V100S GPUs.

To comprehensively evaluate the performance of classification models, we adopt widely used metrics, accuracy (AC), precision (PR), recall (RC), and F1 score (F1).

In our evaluation, precision, recall, and F1 score are macro-averaged to ensure equal consideration of all classes regardless of their frequency.

Additional Evaluation: Malware Dataset

We further investigated the 1% lower F1 score of FlowletFormer versus TrafficFormer on the Malware classification task. Minor variations (like 1% difference) on this dataset can occasionally arise from factors like specific data splits. To this end, we generated 5 random dataset splits by varying the random seed, and found that in 3 cases, FlowletFormer either outperformed or matched TrafficFormer on the Malware dataset. Table 11 reports the F1 scores under each seed.

More Ablation Study

To support the figures in the main text and further illustrate the robustness of our approach, we provide complete numer-

ical results of the ablation study across all eight downstream datasets, as shown in Table 12 and Table 13.

To thoroughly investigate the contribution of each component in **FlowletFormer**, we conducted a series of ablation experiments. The results in Table 12 and Table 13 report the performance of the full model and various degraded versions, where specific modules were removed.

Impact of Flowlet and Field Tokenization (FL). Removing the Flowlet and Field Tokenization module (w/o FL) led to significant performance drops on most datasets. In this variant, the traffic representation and tokenization revert to the burst and BPE tokenization, which is consistent with the approach used in ET-BERT. For example, on the ISCX-Tor2016 dataset, the accuracy decreased from 0.9215 to 0.8328 and the F1-score from 0.9116 to 0.6924. The effect is even more pronounced on the Browser dataset, where accuracy dropped from 0.7050 to 0.3700 and F1-score from 0.6684 to 0.3099. These results highlight the critical role of Flowlet segmentation and field-aware tokenization in capturing temporal dependencies and contextual coherence within sessions. By introducing Flowlets, the model learns to represent traffic in a behavior-aware manner, which facilitates more robust classification of dynamic network flows.

Impact of Masked Field Model (MFM). The removal of the masked field modeling task (w/o MFM) has dataset-specific effects. For instance, on the ISCX-VPN(Service) dataset, accuracy dropped dramatically from 0.9400 to 0.5467, indicating that MFM plays a critical role in modeling datasets with rich and structured protocol field information. It likely helps the model capture inter-field dependencies and learn which fields are important for traffic differentiation. In contrast, datasets like CSTNET-TLS and CIC-IDS2017 showed less degradation, suggesting that those tasks are less sensitive to fine-grained field semantics.

Impact of Flowlet Prediction Task (FPT). Removing the Flowlet Prediction Task (w/o FPT) caused performance degradation across several datasets, though less severe than w/o FL or w/o MFM. For example, in ISCX-Tor2016, accuracy dropped from 0.9215 to 0.9044 and F1-score from 0.9116 to 0.8429. This indicates that FPT serves as an effective auxiliary task, guiding the model to learn patterns in the temporal evolution of traffic flows, which indirectly enhances downstream classification.

Impact of Protocol Stack Alignment-Based Embedding (PE). The removal of the protocol embedding layer (w/o PE) resulted in a consistent but relatively moderate drop across datasets. For instance, F1-scores dropped by 2–4%. This suggests that while PE enhances the model’s ability to capture protocol-layer semantics, it is not the main

Table 10: Overview of Fine-Tuning Tasks and Datasets.

Task	Dataset	Flow Number	Class Number	Label
Service Type Identification	ISCX-VPN (Service)	1,500	6	VPN-Chat,VPN-Email,VPN-Ftp...
	ISCX-Tor2016	2,922	8	Audio, Browsing, Chat...
Application Classification	ISCX-VPN (App)	3,289	10	VPN-Youtube,VPN-Voipbuster,VPN-Vimeo...
Website Fingerprinting	CSTNET-TLS	46,375	120	acm.org.adobe.com.alibaba.com...
Browser Classification	Browser	2,000	4	Chrome,Firefox,Internet,UC
Malware Classification	USTC-TFC	8,000	16	Miuref,FTP,Gmail...
Traffic Classification	CIC-IDS2017	6,000	12	Benign,Botnet,DDoS...
IoT Classification	CIC-IoT2022	4,931	12	Attack_Flood,Idle,Interaction_Audio...

Table 11: F1 scores on the Malware dataset under five random splits.

Seed	TrafficFormer	FlowletFormer
SEED1	0.9753	0.9962
SEED2	0.9900	0.9615
SEED3	0.9764	0.9789
SEED4	0.9740	0.9776
SEED5	0.9788	0.9776

performance bottleneck.

Impact of Pretraining (PT). Eliminating the pretraining stage (w/o PT) caused catastrophic performance degradation on all datasets. For example, on ISCX-VPN(Service), accuracy fell from 0.9400 to 0.5467 and F1-score from 0.9364 to 0.3949. These results emphasize the essential role of pretraining in learning generalizable traffic representations and initializing the model with better parameter priors for downstream tasks.

More Few-shot Analysis

To evaluate the capability of FlowletFormer under data-scarce conditions, we conduct a few-shot learning analysis. The results are reported in Table 14 and Table 15. As shown, FlowletFormer achieves competitive performance under full supervision (100% training data). More importantly, it consistently maintains relatively high F1-scores even when the amount of training data is significantly reduced.

For example, on the ISCX-VPN(Service) dataset, FlowletFormer achieves an F1-score of 0.8106 using only 10% of the training data, significantly outperforming traditional models such as AppScanner and BIND. This indicates the strong generalization ability of FlowletFormer in few-shot settings.

However, on the Browser dataset, the performance of FlowletFormer drops more substantially under limited data, suggesting that the traffic patterns in this dataset are more complex and require more data to learn effectively.

More Clarification of Word Analogies Similarity Analysis

To further clarify the purpose and design of the **Word Analogies Similarity Analysis** in Section 4.6, we emphasize that this experiment is not a classification task, but rather a semantic probing analysis inspired by methodologies from natural language processing.

In NLP, analogical reasoning tasks (e.g., “*king - man + woman ≈ queen*”) are commonly used to evaluate whether pretrained language models capture meaningful token relationships. Following this intuition, we designed an analogous probing task in the context of network traffic to examine the semantic structure of token embeddings learned during pretraining.

Specifically, we selected three well-known HTTP-related port numbers (**80**, **8080**, and **8000**) and analyzed their relative positions in the learned embedding space using cosine similarity. These ports are commonly used for HTTP services and frequently co-occur in real-world traffic, thus forming a semantically coherent group.

Our experimental results show that FlowletFormer captures the semantic similarity between these ports more accurately than baseline models. This suggests that the model has developed a deeper understanding of protocol-layer semantics and is capable of organizing related concepts (e.g., similar ports) in a meaningful embedding space.

More Computational Cost and Complexity

Table 16 reports the full comparison of FlowletFormer against two baseline models (ET-BERT and TrafficFormer) across the three experimental phases: pretraining ($6 \times V100$ GPUs, 200 K steps), fine-tuning ($1 \times V100$ GPU, full epochs), and inference (throughput in samples/sec). All runs were carried out under identical hardware and configuration settings to ensure a fair evaluation of runtime, per-step/epoch granularity, and GPU memory usage.

Limitation

Though FlowletFormer achieves fine-grained behavioral analysis within each flowlet, it still has several limitations.

First, the fixed maximum input length forces us to split long flows into shorter flowlets. While this enables detailed study of intra-flow behaviors, it prevents the model

Table 12: **Ablation study results on ISCXVPN2016, ISCX-Tor2016, and CSTNET-TLS 1.3 datasets.** The abbreviations are explained as follows, FL: Flowlet and Field Tokenization, MFM: Masked Field Model, FPT: Flowlet Prediction Task, PE: Protocol Stack Alignment-Based Embedding Layer and PT: Pre-Training.

Dataset	ISCX-VPN(Service)				ISCX-Tor2016				ISCX-VPN(App)				CSTNET-TLS			
Metric	AC	PR	RC	F1	AC	PR	RC	F1	AC	PR	RC	F1	AC	PR	RC	F1
FlowletFormer	0.9400	0.9471	0.9277	0.9364	0.9215	0.9263	0.9043	0.9116	0.8480	0.8153	0.7641	0.7712	0.8605	0.8578	0.8445	0.8473
w/o FL	0.9133	0.9077	0.8983	0.8995	0.8328	0.6978	0.6892	0.6924	0.7872	0.7555	0.6988	0.7085	0.8025	0.7943	0.7795	0.7820
w/o MFM	0.5467	0.5429	0.5323	0.4830	0.4505	0.1790	0.3300	0.2304	0.8146	0.7604	0.7257	0.7341	0.8051	0.8024	0.7853	0.7886
w/o FPT	0.9133	0.8936	0.9138	0.9010	0.9044	0.8189	0.9114	0.8429	0.8055	0.7370	0.7021	0.7057	0.8499	0.8597	0.8327	0.8372
w/o PE	0.9000	0.9087	0.8656	0.8804	0.9044	0.8428	0.9098	0.8653	0.8359	0.7664	0.7534	0.7522	0.8519	0.8449	0.8363	0.8368
w/o PT	0.5467	0.4278	0.4278	0.3949	0.1706	0.0213	0.1250	0.0364	0.4043	0.2689	0.2678	0.2365	0.7622	0.7602	0.7357	0.7358

Table 13: Ablation study results on Browser, USTC-TFC, CIC-IDS2017, and CIC-IoT2022 datasets.

Dataset	Browser				USTC-TFC				CIC-IDS2017				CIC-IoT2022			
Metric	AC	PR	RC	F1												
FlowletFormer	0.7050	0.7742	0.7050	0.6684	0.9675	0.9713	0.9675	0.9675	0.9183	0.9475	0.9183	0.9079	0.9109	0.8905	0.8866	0.8859
w/o FL	0.3700	0.2787	0.3700	0.3099	0.9600	0.9680	0.9600	0.9598	0.8850	0.8870	0.8850	0.8835	0.8401	0.7881	0.7936	0.7875
w/o MFM	0.6600	0.6006	0.6600	0.5976	0.9650	0.9723	0.9650	0.9653	0.4505	0.1790	0.3300	0.2304	0.8968	0.8506	0.8543	0.8473
w/o FPT	0.6850	0.7932	0.6850	0.6428	0.9663	0.9696	0.9663	0.9658	0.9044	0.8189	0.9114	0.8429	0.9049	0.8765	0.8788	0.8736
w/o PE	0.6800	0.7486	0.6800	0.6745	0.9650	0.9689	0.9650	0.9648	0.9044	0.8428	0.9098	0.8653	0.8988	0.8660	0.8593	0.8587
w/o PT	0.2700	0.3138	0.2700	0.1387	0.9563	0.9680	0.9562	0.9571	0.1706	0.0213	0.1250	0.0364	0.8664	0.8073	0.8174	0.8089

from learning unified patterns over entire long flows, which may be crucial for detecting certain sophisticated or slow-evolving anomalies.

Second, our Field Tokenization treats each protocol field as an independent “word” analogous to treating every single Chinese character as a separate token. Although this captures the finest-grained units, it cannot model semantic entities that span multiple fields. In future work, we could adopt Chinese word segmentation techniques to merge common adjacent fields into higher-level tokens.

Third, because FlowletFormer is based on the BERT architecture, both pretraining and real-time inference demand substantial GPU resources. This high computational and memory overhead may limit deployment in resource-constrained environments or scenarios requiring very high throughput.

Lastly, despite introducing protocol-stack alignment and field-aware pretraining objectives, the internal decision process of FlowletFormer remains difficult to interpret and audit. This lack of transparency can be problematic in high-security settings where explainability and trust are paramount.

Broader Impacts While FlowletFormer can significantly enhance the accuracy of anomaly detection and threat mitigation—thereby contributing to more secure and reliable networks—it also carries potential risks. On the positive side, better traffic classification aids in detecting malicious activities (e.g., DDoS, malware propagation) and supports privacy-preserving analytics by filtering out sensitive flows before further processing. On the negative side, the same techniques could be repurposed for intrusive traffic moni-

toring or profiling of users, raising privacy and ethical concerns. To mitigate such risks, we advocate for transparent deployment policies, strict access controls, and regular audits of model usage.

Table 14: Few-shot Analysis (F1-score) on ISCXVPN2016, ISCX-Tor2016, and CSTNET-TLS 1.3 datasets.

Dataset	ISCX-VPN(Service)				Tor				ISCX-VPN(App)				CSTNET-TLS			
Size	100%	40%	20%	10%	100%	40%	20%	10%	100%	40%	20%	10%	100%	40%	20%	10%
AppScanner	0.8546	0.7512	0.6074	0.5065	0.7848	0.7456	0.6195	0.5401	0.6874	0.4382	0.5320	0.2222	0.7023	0.6416	0.5661	0.4018
BIND	0.7699	0.6625	0.4603	0.3290	0.8439	0.7222	0.5582	0.5269	0.5609	0.3182	0.2003	0.1992	0.4189	0.3558	0.2933	0.2299
CUMUL	0.6884	0.5244	0.3873	0.4511	0.6401	0.5749	0.5252	0.5775	0.4554	0.3081	0.2673	0.1550	0.5493	0.4598	0.3659	0.2982
DF	0.3934	0.3349	0.2596	0.0686	0.5492	0.4850	0.2499	0.1557	0.2289	0.1906	0.1476	0.1234	0.4933	0.2428	0.0449	0.0543
FSNet	0.9051	0.8384	0.7078	0.3931	0.6028	0.5426	0.4080	0.5743	0.4677	0.4795	0.4752	0.2738	0.7311	0.7132	0.6662	0.5946
GraphDApp	0.7429	0.5713	0.6137	0.2762	0.6383	0.5780	0.4622	0.4895	0.4853	0.2427	0.2203	0.1944	0.6890	0.4948	0.4372	0.3303
Beauty	0.5387	0.3635	0.4063	0.4797	0.2251	0.0676	0.0593	0.1571	0.2964	0.2841	0.2911	0.2748	0.2324	0.0799	0.1059	0.0848
ET-BERT	0.8393	0.3980	0.2450	0.2583	0.7453	0.4959	0.3749	0.3512	0.7066	0.6465	0.5728	0.4631	0.7700	0.7039	0.6117	0.4819
TrafficFormer	0.8821	0.6827	0.5595	0.3909	0.7405	0.4989	0.3506	0.3674	0.6962	0.6085	0.5404	0.4320	0.7704	0.7084	0.6277	0.5660
YATC	0.8279	0.0801	0.0721	0.0947	0.7472	0.6587	0.4994	0.0721	0.7254	0.6489	0.5939	0.1805	0.8140	0.7538	0.6375	0.5040
FlowletFormer	0.9364	0.8956	0.7356	0.8106	0.9116	0.7829	0.7166	0.5917	0.7712	0.8009	0.6224	0.5813	0.8473	0.8171	0.7273	0.6249

Table 15: Few-shot Analysis (F1-score) on Browser, USTC-TFC, CIC-IDS2017, and CIC-IoT2022 datasets.

Dataset	Browser				USTC-TFC				CICIDS2017				IoT			
Size	100%	40%	20%	10%	100%	40%	20%	10%	100%	40%	20%	10%	100%	40%	20%	10%
AppScanner	0.5733	0.3756	0.3524	0.1838	0.8976	0.7407	0.6799	0.5733	0.8630	0.8158	0.7924	0.7265	0.8288	0.6925	0.5149	0.4027
BIND	0.5738	0.5288	0.4229	0.1304	0.7115	0.6609	0.7008	0.5653	0.8788	0.8377	0.7673	0.6860	0.6435	0.6428	0.4828	0.3444
CUMUL	0.5207	0.3986	0.3742	0.1500	0.5183	0.4654	0.3753	0.3631	0.6951	0.5602	0.5031	0.4991	0.6687	0.5582	0.5479	0.2113
DF	0.1627	0.2068	0.1399	0.0833	0.3059	0.2949	0.1508	0.1507	0.5940	0.2704	0.2956	0.2007	0.1647	0.1019	0.0382	0.0096
FSNet	0.6410	0.4364	0.4444	0.1852	0.8042	0.6406	0.5563	0.7091	0.8144	0.7558	0.7244	0.5827	0.7804	0.5518	0.6089	0.4857
GraphDApp	0.4912	0.3238	0.2484	0.2875	0.8249	0.7729	0.6429	0.5219	0.8647	0.8266	0.6106	0.6531	0.6767	0.4627	0.3642	0.1766
Beauty	0.1040	0.2456	0.1730	0.0965	0.3796	0.4208	0.4439	0.2026	0.6567	0.4464	0.3457	0.3357	0.0296	0.1290	0.0593	0.0965
ET-BERT	0.3439	0.3616	0.2280	0.2500	0.9666	0.9669	0.9286	0.8950	0.8911	0.8764	0.7346	0.7405	0.8244	0.7349	0.5630	0.4338
TrafficFormer	0.5154	0.1520	0.1645	0.1154	0.9707	0.9703	0.9406	0.9432	0.8785	0.8725	0.7622	0.6918	0.8048	0.7578	0.5437	0.5190
YATC	0.3320	0.4761	0.4176	0.1613	0.9746	0.9480	0.9655	0.9159	0.8959	0.8854	0.6714	0.5902	0.8288	0.7243	0.7665	0.0758
FlowletFormer	0.6684	0.6230	0.6553	0.3095	0.9675	0.9553	0.9457	0.9380	0.9079	0.8997	0.8610	0.8510	0.8859	0.8237	0.8180	0.6152

Table 16: Computational efficiency comparison across pretraining, fine-tuning, and inference.

Phase	Model	GPUs	Time	Unit/Granularity	GPU Memory (GB)
Pretraining	FlowletFormer	6	42 h	75.67 s / 100 steps	28
	ET-BERT	6	41 h	73.87 s / 100 steps	28
	TrafficFormer	6	45 h	82.00 s / 100 steps	28
Fine-tuning	FlowletFormer	1	1,153 s	57.65 s / epoch	17
	ET-BERT	1	1,177 s	58.85 s / epoch	17
	TrafficFormer	1	1,158 s	57.90 s / epoch	17
Inference	FlowletFormer	1	—	150.04 samples/sec	—
	ET-BERT	1	—	148.92 samples/sec	—
	TrafficFormer	1	—	150.45 samples/sec	—