UTFormer: An Ultra-lightweight Transformer for Traffic Classification

Dong Wen, Tianyun Li, Zhuochen Fan*, *Member, IEEE*, Qing Li, *Senior Member, IEEE*, Jie Li, Fa Zhu, Chenglong Li, Athanasios V. Vasilakos, *Senior Member, IEEE*, Tao Li*

***Abstract*—Transformer-based models have shown promise in network traffic classification, but face challenges of architectural and computational redundancy. This paper presents UTFormer, an ultra-lightweight Transformer that addresses dual redundancies in input features and model architecture. We introduce two key innovations: (1) Dynamic payload distillation that automatically identifies critical bytes through attention-driven selection, reducing input features by up to 95% while maintaining high accuracy; (2) Differentiable architecture search formulates Transformer configurations as continuous parameters that can be optimized via stochastic gradient descent, enabling automated search of compact architectures with layer depth ≤ 4 and hidden dimensions ≤ 256.**

Our evaluation on six datasets shows that UTFormer achieves 87.79%-99.99% accuracy while significantly decreasing computational costs by 9.0×-36398.5× and reducing the model size by 44.9×-5433.9× compared to Transformers-based baselines. UTFormer demonstrates excellent architectural efficiency with a minimum 106 KB model size, enabling traffic classification at the speed of 7.55 million packet-per-second in offline evaluations and reaching the throughput of 10 Gbps line-rate under a real deployment environment. It is demonstrated that our work is capable of performing traffic classification with a lightweight model architecture while maintaining high accuracy. The source code is available at: https://github.com/utformer/UTFormer

***Index Terms*—Traffic classification, Transformer, Self-attention, Stochastic gradient descent.**

I. Introduction

TRAFFIC classification serves as a fundamental building block for network systems, enabling critical applications ranging from fine-grained traffic classification to intrusion traffic detection. To address the inherent complexity of modern network traffic, the research community has increasingly adopted machine learning (ML) and deep learning (DL) techniques. Typical approaches leverage decision trees [1], [2], [3], recurrent neural networks (RNNs) [4], [5], [6], and convolutional neural networks (CNNs) [7], [8], [9] to capture spatial and temporal dependencies in traffic data [10], [11], [12]. Considering the success of Transformer models in natural language processing (NLP) [13], [14], recent work has explored Transformer-based architectures as a paradigm for traffic classification. Leveraging the ability of Transformers to model long-range dependencies in network traffic, it has demonstrated state-of-the-art (SOTA) performance in key tasks including encrypted traffic classification and malware behavior detection [15], [16], [17].

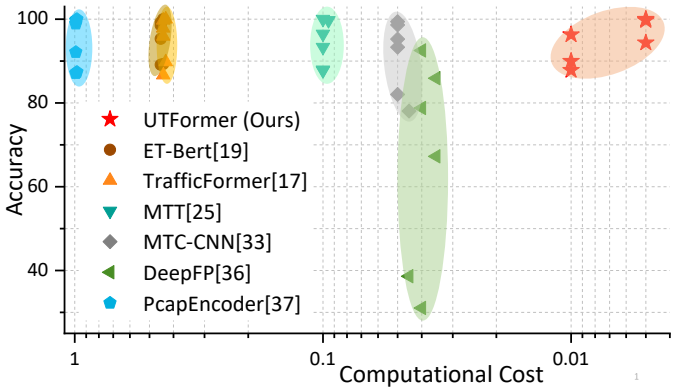


Fig. 1: Accuracy *vs* computational cost. The x-axis (normalized computational cost) decreases left-to-right; the y-axis (accuracy) increases bottom-to-top. Our UTFormer, positioned at the right-top area, exhibits the lowest computational cost and SOTA-comparable accuracy among six baselines.

A. Motivation

Specifically, the emerging Transformer-based traffic classification considers network communication as a contextual language exchange between devices on the network [18]. This novel perspective enables the direct processing of raw packet bytes through tokenization mechanisms without traditional feature engineering pipelines. Then, the self-attention mechanism within the Transformer architecture demonstrates exceptional capability in modeling byte-level inter-dependencies, achieving superior classification performance. However, components such as multi-head self-attention (MHSA) modules and deep Transformer block layers (e.g., the 12-layer architectures in [18], [19]) in existing work raise practical deployment concerns due to their high computational cost. This architecture fundamentally conflicts with the low-latency and high-throughput requirements for traffic classification [20],

Dong Wen, Tianyun Li, Jie Li, Tao Li are with the College of Computer Science and Technology, National University of Defense Technology, Changsha, China. (e-mail: wendong19@nudt.edu.cn; lty_23@nudt.edu.cn; lijie_@nudt.edu.cn; taoli@nudt.edu.cn).

Zhuochen Fan and Qing Li are with the Department of Strategic and Advanced Interdisciplinary Research, Pengcheng Laboratory, Shenzhen 518055, China. (e-mail: fanzc@pku.edu.cn; liq@pcl.ac.cn).

Fa Zhu is with the College of Information Science and Technology & College of Artificial Intelligence, Nanjing Forestry University, China. (e-mail: zhufag@gmail.com)

Chenglong Li is with the Defense Innovation Institute, Academy of Military Sciences, Beijing, China. (e-mail: chenglongli17@163.com).

Athanasios V. Vasilakos is with the Department of Networks and Communications, College of Computer Science and Information Technology, IAU, P.O. Box 1982, Dammam, 31441, Saudi Arabia and the Center for AI Research (CAIR), University of Agder (UiA), Grimstad, Norway (email: th.vasilakos@gmail.com)

* Tao Li and Zhuochen Fan are corresponding authors.

[21]. Therefore, there is an urgent need to design lightweight Transformer models to address such conflicts.

Meanwhile, our ablation experiments on the malware detection task of USTC-TFC dataset [22] reveal that existing Transformer-based traffic classification models have critical redundancy in the model architecture and input feature, as listed in Table I. We select the architecture of ET-Bert [19] as the baseline model (Model-A in Table I), which employs a 12-layer Transformer architecture with 1500-byte of input. For fair comparison, Model-B and Model-C, respectively, have reduced architecture or input features: Model-B has the same input as Model-A but adopts a reduced 5-layer architecture, while Model-C has a shorter input of 512 bytes with the exact architecture of Model-A. To benchmark the computational overhead, we normalize the computation workload of three models against Model-A. (i) **Architectural redundancy.** Although Model-B only requires 42.67% computation compared to Model-A, it maintains comparable classification accuracy , suggesting the architectural redundancy in existing models. (ii) **Input redundancy.** While prior work [23], [24], [25] universally adopts a fixed-length input (256-1500 bytes), Model-C processes only 512 bytes per packet (34.13% of Model-A) while maintaining equivalent accuracy on the same task. The reduction in input features directly reduces the computational overhead by 7.69×, confirming the redundancy of the input feature.

TABLE I: The redundancy in existing Transformer models.

	Model-A	Model-B	Model-C
# layers	12	5	12
Input	1500 B	1500 B	512 B
Accuracy	99.97%	99.99%	99.99%
Normalized computation	1.0	0.42	0.13

B. Challenges and contributions

Despite these empirical observations, designing lightweight Transformer models for traffic classification faces two fundamental challenges: ***(i) How to determine an effective transformer architecture.*** The computational complexity of Transformer models depends on multiple interdependent parameters, including layer depth, self-attention head count, and hidden dimension sizes. These architectural parameters form a high-dimensional combinatorial search space where finding optimal configurations becomes NP-hard [26], [27]. ***(ii) How to select the most informative bytes in the traffic.*** It is difficult to identify the most informative payload bytes without prior knowledge of their spatial distribution, particularly for encrypted traffic classification. There emerges a critical question: How do we reliably locate informative bytes within variable-length network payloads across various protocols and encapsulation formats?

To overcome these challenges, we present ***UTFormer***, an ultra-lightweight Transformer model for traffic classification. Our design introduces two fundamental innovations.

- Dynamic payload distillation: Leveraging the inherent self-attention mechanism of Transformers, we develop an adaptive byte selection technique that automatically identifies and retains semantically significant traffic segments. This *distillation* process is controlled by the differential byte-keeping rate, eliminating tedious manual threshold tuning through gradient-based optimization.
- Differential architecture search: To further determine the efficient architecture of Transformer models, we formulate architectural parameters (layer depth, head dimension, etc.) as continuous optimization variables within a composed search space. A unified loss function guides the simultaneous optimization of model accuracy and computational cost via stochastic gradient descent (SGD). The byte-keeping rate in dynamic payload distillation is also automatically optimized during the search.

Experimental evaluations on six datasets verify the model efficiency and classification performance of UTFormer, as depicted in Figure 1. Compared with the baselines, UTFormer introduces the lowest computational cost with competitive accuracy compared with SOTA models. Moreover, our dynamic payload distillation reduces up to 90% useless traffic bytes, with a differential architecture search technique resulting in 100 KB-level ultra-lightweight Transformer models. In-depth experiments further demonstrate the capacity and potential of dynamic payload distillation to be a drop-in light-weighting method for Transformer-based traffic analysis models.

Our contributions are as follows.

(i) We propose UTFormer, a novel light-weight Transformer model for traffic analysis. With a compact model size of 106KB-12.8MB and 3.61×-36538.46× reduced computational overhead, UTFormer maintains $87.79\%-99.99\%$ classification accuracy on six datasets. Its lightweight design and classification performance make it a practical solution for low-latency and high-throughput traffic classification deployment.

(ii) We present key innovations, including dynamic payload distillation and differential architecture search. These methods co-optimize input features and model architecture, automatically eliminating up to 95% of redundant traffic bytes while constructing ultra-lightweight models. Experiments also prove the compatibility of dynamic payload distillation on other Transformer-based traffic analysis models.

(iii) We conduct rigorous evaluations on UTFormer under real-world deployment environments, and the experimental results demonstrate that UTFormer yields 10 Gbps line-rate throughput with a real-time latency of less than 0.2 milliseconds, proving the practical deployability of UTFormer.

II. Background

A. Transformer models

The Transformer architecture introduces a fundamental structural divergence from traditional deep neural networks (DNNs) through its tokenization paradigm, where Transformer models directly transform raw traffic bytes into a series of tokens. This method originates in NLP, where word tokens enable context-aware processing [13], [14]. In some variants of Transformer models, an extra **class token** [28], [29], [30] is introduced to dynamically aggregate global features through self-attention mechanisms. Classification tasks related to network traffic or computer vision domains can benefit from the use of class token [28], [30].

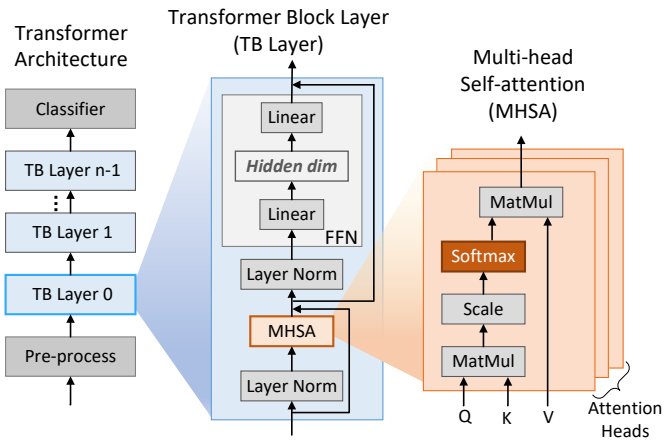


Fig. 2: The architecture of Transformer models.

Transformer architecture comprises multiple stacked Transformer block layers, as illustrated in Figure 2. Each block layer contains two primary components: a Multi-Head Self-Attention (MHSA) module and a Feed-Forward Neural Network (FFN), with each module incorporating a Layer Normalization (`LayerNorm`) operation. The basic building block of the MHSA module is the self-attention mechanism, which projects the input tokens into respective vectors: query (Q), key (K), and value (V) through matrix multiplication (`MatMul`) with three learnable parameter matrices respectively. The attention computation follows the formulation in Equation 1, where l denotes the dimension of token embeddings. The `Softmax` operation generates an **attention matrix** that captures pairwise token dependencies, which subsequently performs weighted aggregation of value vectors through the matrix multiplication with V. A single instantiation of this computation constitutes an attention head, while multiple parallel heads with independent parameters collectively form the complete MHSA module.

The MHSA module is succeeded by an FFN that enhances non-linear representation capacity. As shown in Figure 2, the FFN first expands input token embeddings via a hidden dimension parameter, then projects them back to the original dimensionality. This expansion-contraction design introduces parameterized feature interactions. Obviously, the hidden dimension size of FFN also influences the computational overhead.

$$X = \mathrm{Attn}(Q, K, V) = \mathrm{Softmax}(\frac{QK^T}{\sqrt{l}}) \cdot V \qquad (1)$$

B. Transformer-based Traffic Classification Models

Unlike conventional DL-based traffic classification approaches that rely on hand-crafted statistical features (e.g., flow duration, packet size distributions) [1], [6], [31], [32], Transformer architectures directly process raw packet bytes through end-to-end learning. Such a paradigm has two advantages: (i) The self-attention mechanism automatically discovers contextual patterns in the content of the application layer that conventional aggregated flow characteristics fail to capture, yielding superior classification accuracy across diverse traffic types. (ii) The elimination of manual feature engineering pipelines removes both human biases in feature selection and computational overhead from pre-processing stages, as models operate directly on raw byte sequences without requiring pre-defined statistical abstractions.

Recent advances in traffic classification have seen increasing adoption of Transformer architectures with varying designs of models. MTT framework [25] employs five Transformer block layers with five self-attention heads per layer, processing a 1500-byte packet by aggregating 50 bytes into one token. Subsequent work, such as MTC [33], hybridizes CNN with transformers for enhanced local feature extraction. In contrast, ET-Bert [19] adapts the BERT architecture from the field of NLP, which implements a twelve-layer deep network with twelve attention heads per layer, becoming the most complex traffic classification transformer currently. While these approaches demonstrate 87%-99% classification accuracy across various datasets, they usually prioritize model performance over lightweight efficiency, overlooking critical system constraints like computational complexity, memory footprint, and practical deployment considerations for resource-constrained network environments.

III. Method

A. Overview

UTFormer achieves a lightweight Transformer architecture for traffic classification through two key techniques: Dynamic Payload Distillation and Differential Architecture Search. As illustrated in Figure 3, the framework first applies dynamic payload distillation in the first block layer to adaptively keep important bytes in raw packets, effectively removing redundant computation on less informative bytes. Consequently, the state information of the Transformer (such as accuracy, byte-keeping rate of distillation, and architectural configurations) is sent to the differential architecture search to evaluate the computational cost and loss value of the neural network. Through normalization and co-efficient tuning, these terms are then unified in the loss function, which finally guides the differential architecture search. The latest search result is updated to generate the configurations that determine the architecture of UTFormer. The optimization process terminates automatically when the gradient-based search reaches convergence or the iteration count reaches the predefined budget, delivering a lightweight architecture with low cost and high accuracy. The variants utilized in UTFormer are noted and explained in Table II.

TABLE II: The Notation table of UTFormer

Variants	Definition
N_h	The number of attention heads
N_l	The depth of block layers
d	The hidden dimension of FFN
γ	The byte-keeping rate
P	The configuration of variants, $P = \{N_h, d, N_l, \gamma\}$
N_{token}	The number of tokens
$\mathcal{L}$	The loss function
$\mathcal{L}_v$	The loss value of neural networks
$\mathcal{C}_\mathcal{A}$	The computational cost of self-attention
$\mathcal{C}_\mathcal{F}$	The computational cost of FFN

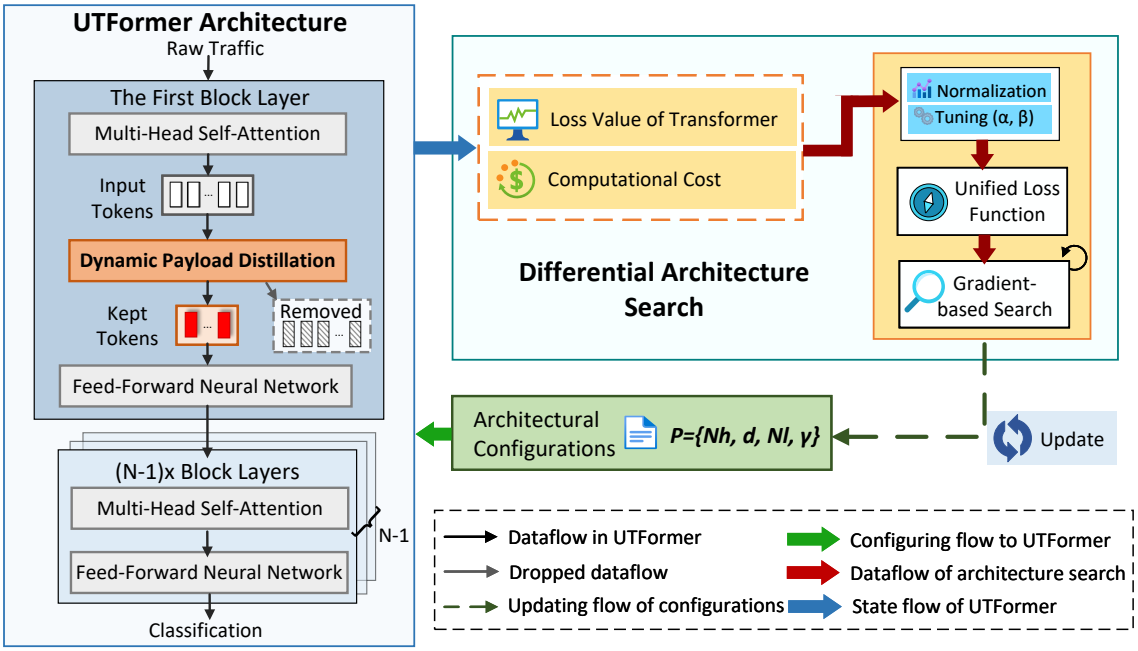


Fig. 3: The workflow of UTFormer, comprising dynamic payload distillation and differential architecture search.

B. Dynamic payload distillation

Modern Transformer-based traffic classification models, such as MTT [25] and ET-BERT [19], employ byte-level tokenization where raw traffic is segmented into fixed-length token sequences. Such a method motivates rephrasing the feature selection challenge: identifying the most useful bytes now reduces to detecting the informative tokens within the self-attention mechanism. Our proposed dynamic payload distillation introduces a novel methodology leveraging the inherent self-attention mechanisms to achieve adaptive byte selection. As mathematically formalized in Equation 2, the computation of the class token (x_{class}) inherently encapsulates this distillation process through the attention-scored aggregation across global tokens, where informative tokens emerge naturally from the attention matrix.

In the computation of self-attention, let q_{class} denote the query vector and x_{class} denote the representation of the class token. The value projection $V = [v_1, v_2, ..., v_n]^T$ constitutes a learnable combination of all n token embeddings, where v_i represents the contextual encoding of i-th token. We denote the vector $\mathbf{s} = [s_1, s_2, ..., s_n]$ generated by $\mathrm{Softmax}$ function in Equation 2. This vector inherently quantifies the cross-token informative confidence due to the global computation of $\mathrm{Softmax}$. When positioned as the initial token, the attention vector $\mathbf{s}$ of the class token corresponds to the first row of the attention matrix. This perspective allows Equation 2 to admit the following reformulation in Equation 3.

$$x_{class} = \mathrm{Softmax}(\frac{q_{class} \cdot K^T}{\sqrt{l}}) \cdot [V_1, V_2, \ldots, V_n]^T \quad (2)$$

$$x_{class} = [s_1, s_2, \ldots, s_n] \cdot [V_1, V_2, \ldots, V_n]^T \quad (3)$$

Formally, the attention vector of the class token $\mathbf{s}$ defines a set of attention confidence governing the inter-token composition, where s_j quantifies the global contribution of the j-th token to the representation of the class token. As established in Section II-A, the class token represents a global feature integration across the input sample. Consequently, $\mathbf{s}$ encodes the confidence relationships between traffic semantics and byte/token-level features within payloads. Higher s_i values indicate a stronger influence between the i-th token and the final classification decision. This attention-driven confidence distribution effectively operates as a dynamic metric, enabling the identification of traffic segments that are most influential to the classification.

Building on our analysis, we propose dynamic payload distillation, as shown in Figure 4. The raw traffic is segmented into discrete tokens, with the class token positioned at the first. After computing the attention matrix through the $\mathrm{Softmax}$ function, we extract its first row $\mathbf{s}$, where s_i represents the attention-based confidence score of the i-th token. Tokens are ranked in descending order by s_i; the Top-K important tokens correspond to the most informative bytes. A tunable byte-keeping rate γ controls the distillation process, which retains the Top-$[\gamma n]$ important tokens while removing low-confidence tokens, n is the number of overall tokens in the Transformer. As discussed in further detail in Section III-C, variant γ is differential, and thereby it can be automatically determined by the differential architecture search.

The entire dynamic payload distillation occurs between the MHSA and FFN modules, reducing the size of input data for the FFN and subsequent layers. By constraining the distillation to the first block layer of UTFormer, we maximize computational efficiency across all downstream operations while preserving critical information. Moreover, the dynamic payload distillation is established on the attention matrix,

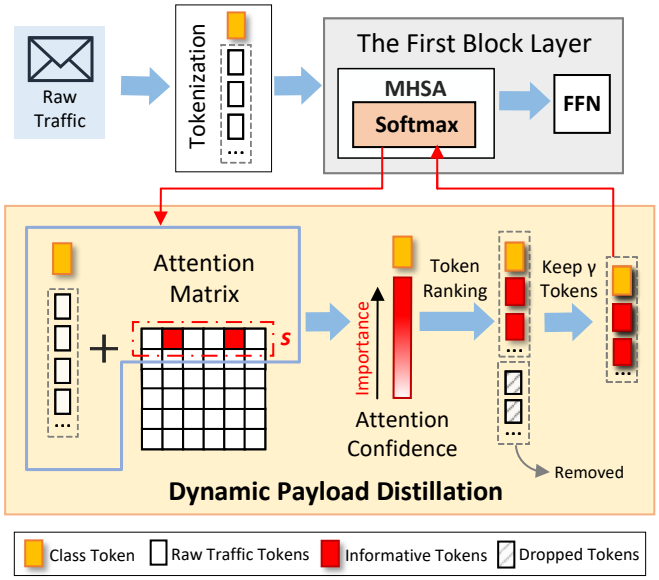


Fig. 4: Dynamic payload distillation.

which is inherently derived from the computing process of the self-attention module. Only a simple Top-K sorting operation is added to the distillation process, introducing no significant overhead to the UTFormer.

C. Differential architecture search

Dynamic payload distillation effectively improves the input redundancy of the input data. To further remove the architectural redundancy of Transformer models, we propose the differential architecture search.

The architectural configuration of UTFormer is governed by four key parameters: the number of attention heads in MHSA modules (N_h), the depth of block layers (N_l), the hidden dimension of FFN (d), and the byte-keeping rate (γ) from the dynamic payload distillation. We formalize these as a multidimensional parameter space $P = \{N_h, d, N_l, \gamma\}$ to jointly optimize the computational overhead and prediction accuracy of the Transformer models. However, these parameters formulate a complex problem space, where manual parameter tuning could be prohibitively expensive due to exponential state-space growth. To address this problem, we implement an automated parameter optimization via stochastic gradient descent (SGD)-based search. This approach automatically navigates the configuration space to identify near-optimal architectures of UTFormer that balance computational efficiency and task accuracy.

Problem space. Stochastic gradient descent operates as an iterative optimization protocol that approximates gradient descent by stochastically sampling gradient directions to minimize a differentiable loss function. This stochastic approximation enables an efficient search of the high-dimensional parameter spaces. However, employing SGD to optimize the architecture of UTFormer faces two critical challenges. (i) *Multiscale parameter coordination:* The architectural parameter space $P = \{N_h, d, N_l, \gamma\}$ contains variables with significantly mismatched numerical characteristics. The byte-keeping rate γ operates in a continuous space $(0, 1]$, while structural parameters N_l and N_h usually occupy small integer domains $\{1, 2, 3, ..., 12\}$. In contrast, the hidden dimension d spans large integer values over 1024. This discrepancy, from 0.01-scale for γ to 1000-scale variations for d, creates unstable gradient interactions during joint optimization. (ii) *Discrete parameters search.* Key architectural parameters (N_h, d, N_l) inherently reside in discrete integer spaces, fundamentally conflicting with the requirement of SGD for continuous differentiable variables. Direct gradient computation becomes infeasible for these integer parameters. This mismatch between discrete search space and continuous optimization mechanics frequently leads to suboptimal architectural convergence.

Methods. To address such problems, we propose the architectural normalization via L1-regularization on P. Each parameter $p^t \in P$ in the t-th iteration is first projected into a unified numerical space through range normalization as in Equation 4. The $\mathrm{Max}(p)$ is the architecture-specific upper bound, e.g., $\mathrm{Max}(N_l) = 12$ aligned with the configurations of ET-Bert, currently the largest Transformer-based traffic classification model. This transformation eliminates magnitude disparities by constraining all parameters to $p_R^t \in (0, 1]$ while preserving relative scaling relationships. Moreover, the architectural normalization via L1-regularization induces an approximate continuous space for originally discrete integer parameters, making theoretical gradients exist across all architectural variables. Consequently, in Equation 5, SGD searches in this approximate continuous space, and parameters are updated along the regularized gradient directions while maintaining architectural validity. After each optimization step, N_l, N_h, and d are mapped back to their native integer domains via rounding operations in Equation 6, completing the current optimization step of UTFormer architecture search.

$$p_R^t = ||p^t|| = \frac{p^t}{\mathrm{Max}(p)} \tag{4}$$

$$p_R^{t+1} = p_N^t - lr \cdot \frac{\partial loss}{\partial p_R^t} \tag{5}$$

$$p^{t+1} = \mathrm{round}(p_R^{t+1}, \mathrm{Max}(p)) \tag{6}$$

Algorithm. The entire SGD-based searching method is illustrated in algorithm 1. The SGD-based search procedure (Algorithm 1) begins by initializing the parameter P and the SGD learning rate lr. A Transformer instance $Model(P)$ is constructed through parametric configuration and trained for multiple epochs, followed by the evaluation of its accuracy via validation loss $loss_v$ (line 2-3). This metric feeds into our proposed loss function, where subtracting the historical loss baseline Ω_{Loss} yields the loss variance $\triangle loss$ for gradient computation (line 4-5). Based on these gradients, we update to regularized parameters P_R (line 6), which are subsequently discretized to their integer form P (line 7-8). The process iteratively refines P until parameter stability or maximum iteration count is reached, ultimately producing near-optimal configurations for UTFormer (line 9).

Algorithm 1 SGD-based searching

Input: A set of initial parameters P, learning rate lr
Output: the final parameters of P

while P is unstable or # iter $\leq$ max iter count **do**
  train the Model(P)
  $loss_v \leftarrow$ Validate(Model(P))
  $Loss \leftarrow$ Loss function($loss_v, P$)
  get $\triangle loss$ && $\Omega_{Loss} \leftarrow Loss$
  $\partial P_R \leftarrow$ get_grad($\triangle Loss, \Omega_{Loss}, lr, P_R$)
  update P_R with ∂P_R
  $P \leftarrow P_R$
end while

D. Unified loss function

Our loss function in Equation 7 mediates the important accuracy-efficiency trade-off in Transformer architecture search. $\mathcal{L}_v(P)$ directly quantifies the model accuracy under architecture configuration P through the model forwarding, making SGD iterations acquire the accuracy performance. The computational cost term $C(P)$ uses floating-point operations (FLOPs) as an architecture complexity metric. For Transformer-based traffic classification models, we derive $C(P)$ by layer-wise FLOP estimation in Equation 8.

$$\mathcal{L} = \alpha||\mathcal{L}_v(P)|| + \beta||\mathcal{C}(P)|| \tag{7}$$

$$\mathcal{C}(P) = \mathcal{C}_{\mathcal{A}}(P) + \gamma[(N_l - 1)\mathcal{C}_{\mathcal{A}}\gamma(P) + N_l\mathcal{C}_{\mathcal{F}}(P)] \tag{8}$$

$$\mathcal{C}_{\mathcal{A}}(P) = 6 \cdot N_h \cdot N_{token} \cdot l^2 + 2 \cdot N_h \cdot N_{token}^2 \cdot l \tag{9}$$

$$\mathcal{C}_{\mathcal{A}}\gamma(P) = 6 \cdot N_h \cdot \gamma N_{token} \cdot l^2 + 2 \cdot N_h \cdot \gamma^2 N_{token}^2 \cdot l \tag{10}$$

$$\mathcal{C}_{\mathcal{F}}(P) = 4 \cdot l \cdot d \cdot \gamma N_{token} \tag{11}$$

Computational cost: The computational cost of Transformer models emerges mainly from MHSA and FFN modules. The dynamic payload distillation is positioned after the first MHSA module, and thus, the subsequent modules only need to process the kept tokens. Therefore, it is necessary to calculate the computation of the first MHSA ($\mathcal{C}_{\mathcal{A}}$) and other MSHA modules ($\mathcal{C}_{\mathcal{A}}\gamma$), respectively. The computational cost of the first MHSA module approximately contains two parts: the matrix multiplication of calculating Q, K, V, and the following operations, such as $\mathrm{Softmax}$. The calculation of two parts is separately demonstrated in Equation 9, where N_{token} denotes the number of tokens (bytes). Similarly, the computation of other MHSA modules can be calculated by Equation 10, with γ controlling the distillation process. This equation reveals that **the number of tokens greatly influences the computational complexity of the Transformer model with a quadratic proportionality**, reflecting the critical importance of dynamic payload distillation for reducing redundant tokens. After each MHSA, FFN mainly utilizes two linear layers for token computing, with d determining its computational complexity. We calculate its computational cost by the equation 11.

Trade-off through co-efficient tuning: The unified loss formulation faces a similar numerical stability challenge where the loss value of Transformer models $\mathcal{L}_v$ typically resides in the $(0, 2]$ range, while the computational cost $\mathcal{C}$ spans nine orders of magnitude (GFLOPs). We address this problem by the normalization method introduced in Section III-C as well. Additionally, we employ two coefficients α and β to explicitly control the accuracy-cost trade-off, where α modulates prediction accuracy and β governs computational overhead. Section V-B quantifies their sensitivity through ablation studies.

IV. Dataset Preparation: Pre-processing & Splitting

UTFormer is evaluated on six different traffic datasets and classification tasks, as demonstrated in Table III.

- ISCX-VPN-2016 (ISCX) datasets [34]: This dataset collects six classes (E-mail and FTP, etc.) of traffic encrypted by VPN. The encrypted traffic classification task is conducted on this dataset to evaluate the UTFormer.
- USTC-TFC (app) datasets [22]: The USTC-TFC (app) dataset includes application-level (app-level) traffic from ten popular applications. Therefore, we perform application-level traffic classification tasks to validate UTFormer using this dataset.
- USTC-TFC (malware) datasets [22]: The USTC-TFC (malware) dataset contains both normal and malware traffic for intrusion detection tasks. Six different types of attack traffic are collected in the dataset.
- Cross-platform-android [35]: The dataset collects encrypted traffic data of the 121 most popular applications on Android. The collected data offers the base of encrypted app-level traffic classification.
- Cross-platform-iOS [35]: The dataset collects encrypted app-level traffic from the 167 most popular applications on iOS. The task in this dataset is encrypted app-level traffic classification as well.
- CampusVideo: This dataset (CapVideo) targets five popular video applications in China (e.g., TikTok and Tencent Meeting) and was collected from the campus network in 2024. To better support application-level traffic classification, CapVideo also includes background traffic from the campus network, reflecting real-world traffic patterns. All traffic in the dataset has been anonymized to preserve privacy.

Following standard practice in traffic classification [25], [17], we adopt a uniform per-packet pre-processing pipeline, including: (1) remove Ethernet headers, (2) mask IP address in network-layer headers, and (3) mask transport-layer port numbers to prevent potential bias from layer-2/3/4 identifiers. Besides, SYN/ACK/FIN packets, and packets of ICMP/ARP protocols are removed. (4) The remaining header fields and payload are concatenated and zero-padded to a fixed 1500-byte length, enabling dynamic payload distillation to have full-sequence attention on any possible valuable bytes. (5) Each byte undergoes a float-32 projection via $x' = x/256$, where 8-bit unsigned range $[0, 255]$ is mapped to the float range $[0, 1]$. For each dataset, we partition the data by flow

TABLE III: The detail of six evaluated datasets, three of them collect encrypted traffic from real-world applications.

Datasets	ISCX-VPN-2016	USTC-TFC (app)	USTC-TFC (malware)	Cross-platform-android	Cross-platform-iOS	CapVideo
Category	Encrypted	App-level	Malware	Encrypted & App-level	Encrypted & App-level	App-level
# Classes	6	10	6	121	167	6
# Train-pkts	3.1M	1.5M	3.5M	299.8K	2.5M	3.1M
# Test-pkts	880.5K	581.7K	1.2M	85.7K	733.3K	1.3M
Volume	20 GB+	3.1 GB	3.9 GB	0.75 GB	2.41 GB	3.31 GB

into training, validation, and test sets: 70% of the flows and their constituent packets are assigned to the training set, 10% to the validation set, and 20% to the test set. **The code for dataset pre-processing and splitting is open-sourced at our homepage for reproduction.**

V. Experimental Setup and Implementation

A. Experimental environment and evaluation metrics

The evaluations of UTFormer and baselines involve two paradigms: the **offline** experiments and the **online end-to-end** experiments, as summarized in Table IV.

Offline evaluation. The offline evaluation encompasses the differential architecture search and training for UTFormer, as well as the reproduction of all baseline models. Each model is validated by two orthogonal dimensions: (1) Traffic analysis capability, measured by classification accuracy (AC), precision (PR), recall (RC), and F1 score (F1); (2) Architectural efficiency, quantified by inference-per-second (IPS) and GPU memory footprint on a server platform.

Online evaluation. The online evaluation accesses the performance of distinct models in real-world deployment environments using the end-to-end test bed shown in Figure 5. The test bed incorporates a traffic tester that replays traffic collected by different datasets to a GPU server through a 10 Gbps NIC. We implement a C-based DPDK program to receive test traffic into a 4GB shared memory buffer, which is accessed by four Python-based threads transforming raw traffic into PyTorch tensors. These traffic tensors are batched every 4096 packets and then moved to the GPU to trigger model inference. We define a model's end-to-end throughput as the maximum input traffic rate it can sustain without causing buffer overflow, and its latency as the end-to-end delay to inference one data batch under deployment conditions.

Hardware and software configurations. Both offline and online experiments are conducted on a system equipped with an NVIDIA RTX 4090 GPU (48 GB GPU memory), an Intel Core i9-14900K CPU, and 80 GB of DRAM, running Ubuntu 20.04 with Numpy 1.24, PyTorch 1.13.1, CUDA 11.6, and cuDNN 8. In the online evaluation, the test bed employs an Intel 82599ES 10 GbE NIC and a TOYO Corporation SYNESIS traffic tester to replay traffic from pcap files and receive traffic at line rate (10 Gbps). The DPDK version is 22.11.

TABLE IV: The offline and online experiments

Paradigm	Tasks	Evaluation metrics
Offline	Train & search UTFormer, Reproducing baselines, Test architectural efficiency.	Accuracy: AC, PR, RC, F1, Efficiency: inference/sec and memory footprint.
Online	Test end-to-end throughput and real-time latency.	Throughput: bits/sec (bps), Latency: millisecond (ms).

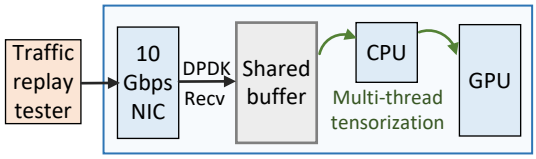


Fig. 5: Online end-to-end test bed.

B. Implementation

The implementation of UTFormer includes the differential architecture search phase and the final model training, with distinct configurations aligned to each stage.

Configurations of the differential architecture search. We constrain architectural parameters $P = (N_h, N_l, d, \gamma)$ with maximum thresholds $N_h^{max} = 12, N_l^{max} = 12, d^{max} = 1024$. Such configuration is inherited from ET-Bert [19] (12 self-attention heads and 12 block layers). We enforce $\gamma >= 0.05$ to prevent catastrophic information loss while preserving at least five tokens (one class token plus four content tokens). With the learning rate=0.005, each SGD iteration includes four model training epochs for stable loss estimation.

The configurations of the unified loss function. The unified loss function balances accuracy and computational cost by setting ($\alpha = 0.5, \beta = 1.5$). The computational cost regularization uses the cost of ET-Bert, the largest Transformer-based traffic classification model, as the regularization upper bound for architecture search. The SGD searching terminates with 15 iterations or early-stopping when the loss function becomes stable.

The configurations of the model training. We set batch-size as 256 across all datasets, with the use of the Adam optimizer and a learning rate = 0.001. Once the architecture of UTFormer is determined, we further train the model with another 40 epochs using the cross-entropy loss function.

C. Baseline models and reproduction

We evaluate UTFormer against multiple representative DL-based models. The baseline models span (1) Light-weight CNN architectures with model size smaller than 20 MB, (2) Transformer models with model size ranging from 40 MB to 576MB, and (3) The latest large-language-model-based approach requiring over 1 GB parameters. (4) The latest Mamba-based model, which is established on state space model (SSM). The architectural detail of all baseline models is listed in Table V.

- MTC-CNN [33] is a light-weight CNN model implementing three 1D convolutional layers for raw packet processing. This model is selected to directly compare the computational efficiency with UTFormer.

TABLE V: The architectural comparison between UTFormer and baselines.

Model	Model type	# Layer	# Head	Hidden dimension	Byte-keeping rate	Model size	Computation cost
MTC-CNN[33]	Light-weight CNN	15	-	-	-	19.5 MB	521.3 MFLOPs
DeepFP[36]	Light-weight CNN	12	-	-	-	15.7 MB	444.2 MFLOPs
MTT[25]	Transformer	5	5	1024	-	40.6 MB	1.1 GFLOPs
ET-Bert[19]	Transformer	12	12	768	-	576 MB	5.7 GFLOPs
TrafficFormer[17]	Transformer	12	12	768	-	576 MB	5.7 GFLOPs
PcapEncoder[37]	LLM-based	24	24	768	-	1210 MB	11.8 GFLOPs
NetMamba+[38]	Mamba	6	-	-	-	27 MB	694 MFLOPs
UTF-ISCX	Light-weight Transformer	1	4	268	5%	**156 KB**	**241.8 kFLOPs**
UTF-USTC-app	Light-weight Transformer	1	1	64	5%	**106 KB**	**156.6 kFLOPs**
UTF-USTC-mal	Light-weight Transformer	1	1	72	5%	**107 KB**	**160.0 kFLOPs**
UTF-android	Light-weight Transformer	4	4	224	25%	**8.8 MB**	**122.8 MFLOPs**
UTF-iOS	Light-weight Transformer	4	4	256	15%	**12.8 MB**	**87.3 MFLOPs**
UTF-CapVideo	Light-weight Transformer	1	1	64	10%	**106 KB**	**180.4 kFLOPs**

- DeepFP [36] is a light-weight CNN model that utilizes traffic features such as the size and direction information of the traffic. These features are organized into a 2D feature map for CNN inference. Choosing this model demonstrates the comparison between the feature-engineering-based model and end-to-end learning.
- MTT [25] is a vanilla transformer with five layers and five self-attention heads per layer.
- ET-Bert basically follows the architecture of Bert [13] (comprises 12 block layers), and each layer includes 12 self-attention heads with a hidden dimension of 768.
- TrafficFormer [17] is a derivative of ET-Bert and achieves state-of-the-art (SOTA) accuracy by improving training methods.
- PcapEncoder [37] is a pretrained model based on T5 [39], an open-source large language model (LLM). The LLM-based PcapEncoder is the largest model in the field of traffic classification, demonstrating generalization to different traffic datasets.
- NetMamba+ [38] is built on SSM-based Mamba model, which combines the high accuracy of Transformer models and the low-overhead of RNNs.

MTC-CNN, MTT, ET-Bert, PcapEncoder, and NetMamba+ provide open-sourced pretrained parameters for their backbone neural network (NN). As PcapEncoder suggested, we reuse these backbone parameters and keep them frozen, only adding and training a dataset-adaptive classifier layer following the backbone. Meanwhile, DeepFP and TrafficFormer do not release pretrained parameters; therefore, we train both models entirely from scratch. In offline experiments, we use a batch size of 256 and a learning rate of 0.0001 to train UTFormer and reproduce all baselines for 30 epochs during the offline experiments, as detailed in Section V-A.

VI. Evaluation

Evaluation guideline: In Section VI-A, we introduce the architectural configurations of UTFormer with a comparison of baseline models. Consequently, we conduct comprehensive end-to-end comparative experiments on accuracy performance (in Section VI-B), practical throughput (in Section VI-C). In the Ablation Study Section, we thoroughly explore the mechanism of the proposed differential architecture search (Section VII-A) and dynamic payload distillation (Section VII-B). Several experiments are designed to validate the convergence performance of the differential architecture search. Simultaneously, the deep dive reveals the position of the most informative bytes within the payload, and dynamic payload distillation can potentially be an off-the-shelf lightweight technique for Transformer-based traffic classification models.

A. The architecture of UTFormer

In Table V, we present the architectural configurations determined by the differential architecture search . The resultant models are denoted by the namesake of respective datasets, as listed in Table V (UTF-ISCX, UTF-USTC-app, UTF-USTC-mal, UTF-android, UTF-iOS, and UTF-CapVideo).

Architectural Efficiency: UTFormer exhibits outstanding light-weight capacity across different datasets. On ISCX-VPN, USTC-TFC (app), USTC-TFC (mal), and CapVideo datasets, the search converges to a single-layer architecture, maintaining **100KB-level model size**. Dynamic payload distillation further eliminates 90%-95% of input bytes, yielding computation below 200 kFLOPs. On Cross-platform-android and Cross-platform-iOS datasets, UTFormer adapts to multi-class challenges with a configuration of 4 block layers $\times$ 4 heads, expanding model size to MB-level. Despite the parameter growth, the distillation removes 75%-85% of input, saving over 9.0$\times$ computational cost.

Architectural comparison. UTFormer exhibits significant architectural efficiency improvements over CNN baselines. UTFormer achieves 1.2-1844.3$\times$ reduction in model size compared to MTC-CNN and DeepFP, reducing computational cost by 3.6-3328.9$\times$. In contrast to Transformer baselines, UTFormer removes redundancy in both model architecture and input. Existing Transformers like ET-Bert [17], [19] directly adopt Bert-based architectures [13] with fixed-length payload inputs (e.g, 1500B in MTT [25]), leading to excessive model size up to 576 MB and computation overhead over 1 GFLOPs. Our evaluation shows that UTFormer reduces the model size by 44.9-5433.9$\times$ while eliminating redundant computation by 9.0-36398.5$\times$. When compared to PcapEncoder, a LLM-based traffic analysis model, the model size of UTFormer is up to five orders of magnitude smaller, and the computational cost is reduced by up to four orders of magnitude. Moreover, UTFormer still delivers 2.11-254.7$\times$ smaller model size than NetMamba+, and further reduces 5.65-4448.7$\times$ computation.

TABLE VI: The accuracy results across distinct datasets. The highest value of accuracy is highlighted in red, and the second-highest value is in yellow. The difference in parentheses represents the gap between UTFormer and the best accuracy.

Datasets	USTC-TFC (app)				USTC-TFC (mal)				ISCX-VPN-2016			
Metrics (%)	RC	PR	F1	AC	RC	PR	F1	AC	RC	PR	F1	AC
MTC-CNN[33]	99.41	99.74	99.53	99.40	98.52	98.65	98.62	98.60	95.75	91.94	92.92	93.31
DeepFP[36]	78.83	78.19	78.59	78.77	85.09	86.33	85.74	85.88	68.45	66.78	67.81	67.23
MTT[25]	99.97	99.68	99.88	99.68	99.98	99.88	99.95	99.97	95.75	91.94	92.92	93.31
ET-Bert[19]	99.15	99.16	99.15	99.16	99.99	99.99	99.99	99.99	94.39	97.31	95.33	95.29
TrafficFormer[17]	99.80	99.21	99.50	99.71	99.99	99.99	99.99	99.99	96.21	94.50	95.80	95.89
PcapEncoder[37]	97.28	97.05	97.10	99.71	100	100	100	100	91.03	88.24	89.85	92.12
NetMamba+[38]	97.68	97.50	97.65	97.65	99.92	99.99	99.95	99.95	94.50	92.55	94.20	94.60
UTFormer	99.78	99.66	99.74	**99.69** (-0.02)	99.99	99.99	99.99	**99.99** (-0.01)	94.02	91.01	93.24	94.35 (-1.54)

Datasets	Cross-platform-android				Cross-platform-iOS				CapVideo			
Metrics (%)	RC	PR	F1	AC	RC	PR	F1	AC	RC	PR	F1	AC
MTC-CNN[33]	79.63	78.72	80.34	82.04	70.04	75.67	71.38	78.05	95.45	96.03	95.50	95.24
DeepFP[36]	22.32	21.79	21.40	31.06	25.95	26.20	25.27	38.62	93.97	94.60	94.27	94.50
MTT[25]	76.28	75.91	75.50	87.72	86.21	86.11	85.84	87.89	96.35	98.58	97.61	96.44
ET-Bert[19]	86.39	83.19	84.06	89.28	87.57	87.72	87.54	89.10	99.55	96.41	98.88	98.50
TrafficFormer[17]	74.35	72.04	71.67	86.64	89.66	86.97	88.89	89.97	96.18	97.33	97.23	97.83
PcapEncoder[37]	78.06	76.51	77.30	87.36	80.54	89.99	80.63	87.10	98.99	96.55	98.86	98.89
NetMamba+[38]	86.14	86.48	87.23	87.03	88.11	88.21	89.92	89.96	97.65	98.08	97.70	97.71
UTFormer	86.70	85.33	86.29	87.79 (-1.49)	87.96	87.01	87.63	89.87 (-0.10)	97.23	95.71	96.21	96.88 (-2.01)

B. Accuracy performance

Table VI presents the comparison of classification accuracy across six datasets. UTFormer is compared against diverse baselines.

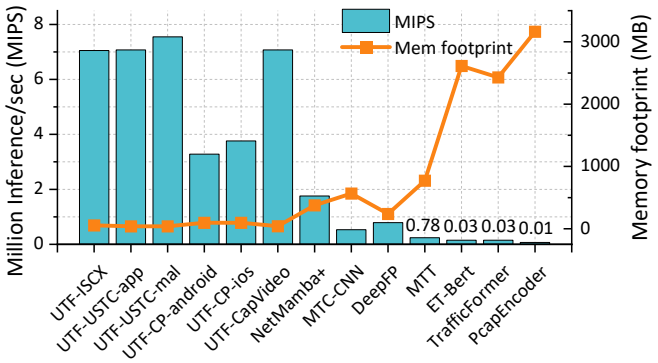
UTFormer vs light-weight CNN and Mamba baselines. As shown in Table VI, our transformer architecture achieves 0.29%-11.82% higher accuracy than MTC-CNN and 14.11%-56.73% absolute accuracy gains over DeepFP across all evaluation scenarios. DeepFP exhibits an unsatisfactory accuracy (38.2%-52.1%), corroborating prior findings [19], [25] about the limitations of feature-based models in handling complex traffic patterns. Moreover, the overall comparison results demonstrate that CNN models fail on accuracy-computational cost efficiency, with a non-negligible gap where UTFormer outperforms CNNs with higher accuracy and lower cost. This efficiency gap stems from structural inefficiencies in CNNs architectures: these CNNs models employ multi-layer convolutional stacks and process full-length sequence inputs, leading to parameter redundancy and unsatisfied computational cost. Our findings align with prior studies [18], [19] that have identified CNNs as suboptimal for traffic classification tasks, which means CNNs require a larger model to achieve comparable accuracy to UTFormer. Based on the architectures of Mamaba models, NetMamba+ achieves accuracy comparable to UTFormer. However, linear feature processing methods make NetMamba+ to occupy 2.11× more parameters for high accuracy, indicating a lower parameter efficiency.

UTFormer vs Transformer and LLM-based baselines. Across distinct datasets, PcapEncoder exhibits the best accuracy in most cases, demonstrating the performance and generalization of the LLM-based approach. Meanwhile, with UTFormer achieving second-place rankings in three datasets, our experimental evaluation reveals that UTFormer consistently achieves top-tier accuracy across multiple datasets and traffic analysis tasks, often matching the performance of much heavier models such as PcapEncoder. Although minor gaps exist in some cases (e.g., 1.54% lower than TrafficFormer on the ISCX-VPN dataset), these are worth it relative to the gains in architectural efficiency. Moreover, all Transformer architectures (including UTFormer) consistently achieve high accuracy across six diverse datasets, confirming their architectural superiority for traffic classification. Transformer models average exceed CNNs by 1.04%-51.17% on accuracy.

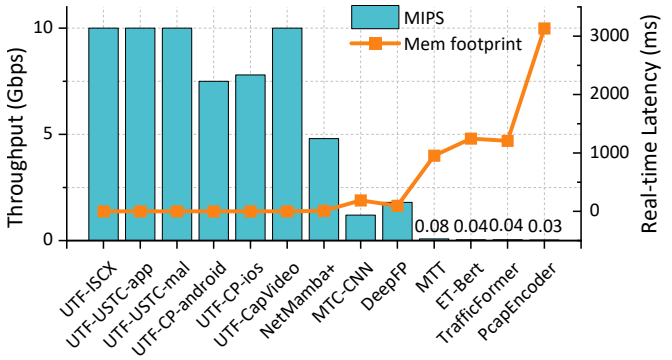
C. Inference performance

As illustrated in Figure 6, further evaluations are conducted in both offline and online environments to compare UTFormer to other baselines.

Offline evaluations. To validate inference performance under offline settings, we use a batch size of 512 for all models to report metrics of inference-per-second (IPS) and memory footprint. The results are depicted in Figure 6a. Due to its large model size and high computational cost, the LLM-based PcapEncoder occupies over 3GB of GPU memory and achieves only 0.07 million IPS (MIPS) in the experiments. Compared to light-weight CNN approaches, Transformer baselines generally demand more memory and deliver lower IPS performance. However, CNN models fall short on accuracy metrics. NetMamba+ delivers higher IPS performance and lower memory footprint than other CNNs and Transformer models due to lightweight SSM architecture. Meanwhile, our experiments demonstrate that UTFormer achieves competitive accuracy as well as strong inference performance. Due to its lightweight architecture, UTFormer performs 3.28−7.55 MIPS performance across different datasets, achieving 107.85× speedup over LLM-based PcapEncoder, 50.33× higher throughput than other Transformers, and 14.25× speedup over CNN baselines. Moreover, the memory efficiency of UTFormer is substantial, only requiring 40-96 MB of GPU memory space, which is 2.48-14.13× smaller than CNNs, and 41.56× smaller than Transformer and LLM-based baselines on average.



(a) Evaluating inference performance and memory footprint in **Offline** experiments.



(b) Evaluating real-time throughput and inference latency in **Online** experiments.

Fig. 6: Inference performance.

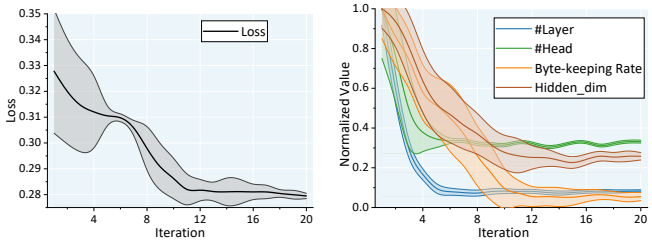
Online evaluations. The online evaluations are conducted on our real-world test bed, as described in Section V-A. The traffic tester generates replay traffic to the GPU server to validate the deployability of different traffic analysis models. We use two key metrics: (1) end-to-end throughput, defined as the maximum input traffic rate a model can sustain without causing buffer overflow, and (2) inference latency, defined as the end-to-end delay to process one data batch. As depicted in Figure 8b, LLM-based and other Transformer baselines deliver throughput lower than 100 Mbps and end-to-end inference latency exceeding 1 second, making them impractical to deploy in a real-world traffic analysis environment due to the performance bottlenecks. Two CNN baselines attain throughput of 1.2-1.8 Gbps with 96-954 milliseconds (ms) inference latency. However, their deployment is still limited by inferior accuracy. Simultaneously, NetMamba+ achieves 4.8 Gbps throughput and 0.08 ms inference latency in online evaluation, demonstrating the lightweight advantage of SSM architecture. In contrast, UTFormer demonstrates superior performance and practical deployability. It reaches 10 Gbps line-rate throughput across four datasets, with a minimum inference latency of 0.11 ms. UTFormer outperforms Transformer and LLM-based baselines by 125.0-333.3× on throughput and up to 28481.8× on inference latency. The experiments conclusively demonstrate the capacity and potential performance of UTFormer for real-world deployment in high-speed traffic analysis tasks.

VII. Ablation Study and Deep Dive

A. Evaluation on differential architecture search

Setup. As discussed in Section V-B, our differential architecture search constrains parameters $P = (N_h, N_l, d, \gamma)$ with maximum thresholds $N_h^{max} = 12, N_l^{max} = 12, d^{max} = 1024$. The byte-keeping rate $\gamma \in [0.05, 1]$ preserves at least five tokens to prevent catastrophic information loss. Each search iteration performs four training epochs for stable loss estimation, guided by a unified loss function balancing accuracy and efficiency (α=0.5, β=1.5). The search terminates at 15 iterations or when parameters stabilize through early stopping.

Convergence performance. Figure 7 captures this optimization trajectory on the ISCX-VPN dataset, showing how the loss function progressively leads to an ultra-lightweight architecture. Figure 7a reveals that the differential architecture search completes convergence within 12 iterations despite a conservative 15-iteration search maximum. To analyze the convergence process in detail, we depict the trajectory of all variants of P in Figure 7b. Two architectural parameters N_l (the number of layers) and N_h (The number of self-attention heads) reach convergence at the 5th iteration. Their relatively small search space ($N_h, N_l \in \{1, 2, 3, ..., 12\}$) ensures a rapid convergence speed. On the other hand, γ (byte-keeping rate) and d (the hidden dimension) convergence around 10-12th iterations due to larger search space (e.g., $d \in \{1, 2, ..., 1024\}$). All relevant variants in P are stabilized within 12 iterations, enabling fast convergence.



(a) The convergence process of loss value.

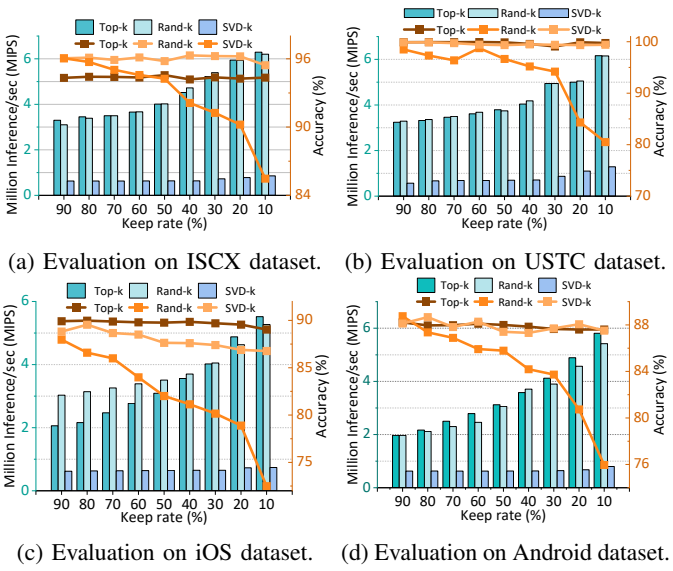
(b) The convergence process of variants in P.

Fig. 7: The convergence process of differential architecture search. The convergence is completed within 12 iterations.

TABLE VII: Model searched by different coefficient choices. From right to the left, loss function weights more on reducing computational cost.

$(\alpha : \beta)$	3: 1	2: 1	1: 1	0.5: 1
Model size	4.6MB	2.2 MB	744 KB	156 KB
byte-keeping rate	42%	24%	10%	5%
Computation (FLOPs)	63.9M	30.7M	2.6M	241.8K
Accuracy	95.01%	94.81%	94.69%	94.35%

Discussion on co-efficient choices of loss function. To validate the impact of different coefficient choices in the unified loss function, we design four groups of experiments with ($\alpha = 3$, $\beta = 1$), ($\alpha = 2$, $\beta = 1$), ($\alpha = 1$, $\beta = 1$), and the configuration ($\alpha = 0.5$, $\beta = 1$) adopted in UTFormer. Loss function with larger α weighs more on the accuracy



(a) Evaluation on ISCX dataset. (b) Evaluation on USTC dataset.

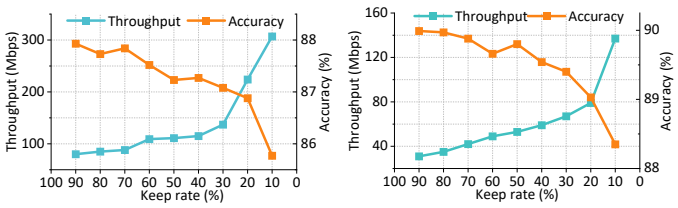
(c) Evaluation on iOS dataset. (d) Evaluation on Android dataset.

Fig. 8: Ablation experiments on Rand-k and SVD-k.

term and less on the cost term, guiding the complicated trade-off of the architecture design. The evaluation experiments are conducted on the ISCX-VPN dataset. For simplicity, we list the ratio between α and β in Table VII. Our empirical experiments reveal that the choice of coefficient value has a significant influence on the accuracy and computational overhead. With $\alpha : \beta$ ranging from 3:1 to 0.5:1, the overall computational cost is reduced from 63.9 MFLOPs to 241.8 kFLOPs, while the accuracy is decreased by 0.74%. Although a smaller α brings a certain influence to the accuracy, its overall reduction in cost is still worth it. As listed in Table VII, the resultant architecture searched with $(\alpha = 0.5, \beta = 1)$ reduces 97.2% model size and 99.6% redundant computation with an acceptable accuracy loss; therefore, we utilize such configurations in determining the architecture of UTFormer. It is also observed that the byte-keeping rate γ decreases from 42% to 5%, demonstrating that the differential architecture search effectively integrates the dynamic payload distillation method, enabling the automated parameter adjustment. The above experiments reveal the parametric sensitivity of the proposed loss function, users can strategically tune (α, β) to adjust the accuracy-cost trade-off based on deployment requirements.

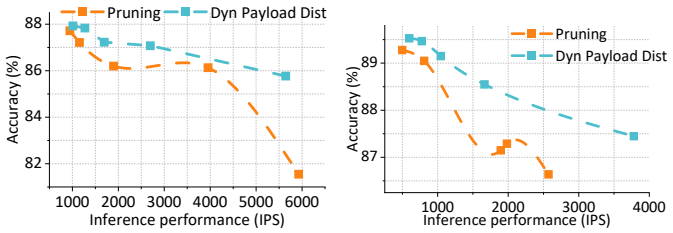
B. Deep dive on dynamic payload distillation

Ablation experiments on Rand-k and SVD-k. To rigorously assess the efficiency of dynamic payload distillation based on Top-k strategy, we perform comprehensive comparisons to (1) random distillation that follows a naive random-k strategy (Rand-k), and (2) singular value decomposition based distillation (SVD-k), which performs SVD operations on the feature matrix X and selects informative tokens based on the sorting results of Σ matrix, as shown in Equation 12. Based on the diagonal matrix Σ, the SVD-k strategy selects k tokens with the largest singular value. We evaluate all methods on four traffic classification datasets—ISCX, USTC-TFC-app (USTC), CP-iOS (iOS), and CP-Android



(a) MTT integrated with dynamic payload distillation. (b) PcapEncoder integrated with dynamic payload distillation.

Fig. 9: Integrating dynamic payload distillation into other Transformer-based and LLM-based traffic analysis models.



(a) Comparing dynamic payload distillation to structured pruning on MTT model. (b) Comparing dynamic payload distillation to structured pruning on ET-Bert model.

Fig. 10: Comparing dynamic payload distillation to structured pruning on Transformer-based baselines.

(Android)—progressively reducing the byte-keeping rate from 90% to 10%.

$$X = U\Sigma V^T \tag{12}$$

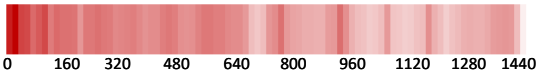
The results in Figure 8 demonstrate that the Rand-k achieves inference performance comparable to the proposed Top-k method at higher keeping rates. However, as the keeping rate decreases, Rand-k suffers severe accuracy degradation due to uninformed and random token selection. In contrast, SVD-k attains high accuracy on par with dynamic payload distillation based on Top-k, confirming that selecting tokens associated with dominant singular values effectively preserves discriminative information. Nevertheless, SVD-k incurs substantial computational overhead from performing SVD on the feature matrix, consuming significant compute resources and offsetting the benefits of token reduction. By contrast, our dynamic payload distillation utilizes the attention matrix, which is already computed as part of the self-attention mechanism in Transformers. The method only adds to Transformers with a Top-k sorting. The method introduces only a lightweight Top-k sorting operation. Given that typical traffic classification inputs contain fewer than 500 tokens [19], [25], this additional overhead is negligible during inference. To conclude, the Top-k-based dynamic payload distillation effectively preserves accuracy while improving the inference performance, achieving a favorable trade-off between accuracy and computational cost.

Dynamic payload distillation as a drop-in technology for Transformer-based traffic analysis. To validate the potential of dynamic payload distillation as a drop-in light-weighting method for Transformer traffic analysis models, we integrate it as a plug-in module into the first layer of both MTT and the LLM-based PcapEncoder. Resulting models are trained on the CP-iOS dataset, and validated on our online test bed. Figure 9a

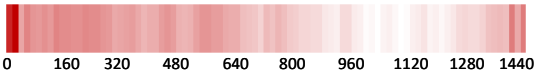
demonstrates the compatibility of dynamic payload distillation with Transformer-based MTT models. At a 20% byte-keeping rate, it improves end-to-end throughput from 80 Mbps to 224 Mbps by 2.8× at the cost of 1.05% accuracy loss. When integrated into the LLM-based PcapEncoder, the presented approach yields a 4.42× end-to-end throughput improvement, with an accuracy degradation of less than 1.5% in Figure 9b. By selectively retaining only the most informative tokens, dynamic payload distillation preserves model accuracy while delivering significant acceleration across diverse Transformer-based traffic analysis models.

Comparison to structured pruning. Although dynamic payload distillation can be seen as an orthometric optimization method relative to general NN light-weighting methods, we still compare the performance of the proposed method against structured pruning [40], [41], [42]. Structured pruning is a GPU-friendly method utilized to reduce model size and computational cost by pruning redundant self-attention heads, FFN channels, or other Transformer components without introducing sparse computation. We reproduce the structured pruning method from [42], progressively removing unimportant self-attention heads within ET-Bert and MTT, followed by fine-tuning both models on the CP-iOS dataset. For example, MTT contains five self-attention heads; we remove these structures and their corresponding parameter channels one at a time, thereby gradually reducing model size and computational cost. Simultaneously, we progressively adjust the byte-keeping rate of dynamic payload distillation to enable a fair comparison with structured pruning. The comparison results in Figure 10 demonstrate that when only a few Transformer attention heads are pruned, both methods achieve comparable accuracy and inference performance. However, as more parameters are reduced, structured pruning endures a significant accuracy drop due to the loss of information carried by the pruned attention heads. In contrast, dynamic payload distillation maintains better accuracy and inference performance, proving its effectiveness as a light-weighting method for Transformer-based traffic analysis.

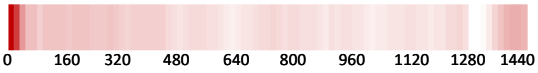
Revealing the most important bytes. Via the dynamic payload distillation method, we try to identify where the most important bytes are in a packet. As detailed in Section V-B, we split the raw bytes into 90 tokens with 16 bytes in each. The dynamic payload distillation method utilizes the attention matrix to choose the most informative tokens (bytes). Therefore, we visualize the confidence value as heat maps from the first row of the attention matrix in Figure 11. The visualization reveals two distinct regions of significance: a primary critical segment spanning tokens 1-6 (bytes 1-96) and a secondary hot spot at tailed tokens, for example, around bytes 800 and bytes 960 in the ISCX-VPN dataset in Figure 11a. The importance of bytes 1-96 aligns with fundamental protocol structures where header fields typically contain transport-layer specifications and application-layer identifiers that are crucial for traffic classification. The secondary hot spot emerges from empirical observations: 12.7% of packets in ISCX-VPN datasets end with 800-960 bytes. With some application-layer protocols (e.g., HTTP and SMTP) appending session identifiers in final payload segments, such a tail segment is significant for traffic classification. Similar phenomena exist in CP-android and CP-iOS datasets as well, as shown in Figure 11b and Figure 11c. Conversely, dynamic payload distillation consistently neglects mid-packet tokens, which typically contain user payloads or generic application data. By focusing computational resources on the identified critical regions, our distillation method achieves substantial computational reduction while maintaining classification accuracy.



(a) The heatmap of ISCX-VPN dataset.



(b) The heatmap of Cross-platform-android dataset.



(c) The heatmap of Cross-platform-iOS dataset.

Fig. 11: The heatmap of raw traffic in datasets. Underlying numbers indicate the location of 0-1500 bytes in a packet.

VIII. Discussion

A. UTFormer lightweight architecture

The performance of UTFormer makes us wonder why UTFormer can be lightweight and high-accuracy. We conclude three reasons as follows:

(i) Dynamic payload distillation removes noise bytes. As illustrated in Section VII-B, the distillation keeps the content near the packet header and packet tail, removing left input tokens. Most removed bytes are user data, which contribute little to the final classification and can be seen as noise. Dynamic payload distillation removes these noises and only keeps the most informative bytes, resulting in reduced computational cost and high accuracy.

(ii) We set no important constraint on the differential architecture search process, yielding the ultra-lightweight UTFormer configuration. Although contemporary Transformer adaptations for traffic classification (MTT[25], ET-Bert[19]) simply inherit the multi-layer paradigms from NLP/CV domains, we fundamentally re-examined architectural prerequisites by mandating only a single attention head and layer as minimal building blocks, exploring the conventional assumptions about depth requirements.

(iii) The problem space of traffic classification is smaller than that in CV and NLP, thus an ultra-lightweight Transformer model can achieve high accuracy. The input of packet-based traffic classification tasks is usually 1500 raw bytes. However, the input data of Transformers can be a 4K high-resolution image, up to dozens of millions of bytes in CV tasks. On the other hand, the output of traffic classification is to classify hundreds of traffic categories, while in large language models, the output space of Transformer models is to generate context content for humans. The problem space of traffic classification suggests that a lightweight Transformer architecture may be sufficient for high performance.

B. Adapting to open-world scenarios

Traffic classification tasks in open-world scenarios could be complex, particularly when traffic and its source application are unseen during training. Currently, numerous works have investigated how to address open-world traffic classification, e.g., leveraging contrastive learning[10], unsupervised learning[15], or open-set method[7] to automatically recognize and label unknown traffic. We want to emphasize that the lightweight technology of UTFormer is orthogonal to the above methods. Therefore, UTFormer can be seamlessly integrated with these technologies to adapt to open-world traffic classification scenarios.

IX. Related work

A. DL-based traffic classification models.

DL-based traffic classification models can be categorized into feature-based and raw-byte-based depending on the input:

Feature-based DL models. Feature-based NN models [43], [44] rely on statistical traffic features as input. Features such as packet arrival intervals, etc., reflect the high-level state of the traffic. However, as revealed by previous research and comparison in our evaluation, feature-based DL models are inferior to raw-byte-based models in accuracy performance.

Raw-byte-based DL models. Raw-byte-based methods [45], [46] feed the models with the raw packet, utilizing important information about the protocol and data distribution carried in the packet, leading to the SOTA accuracy [17], [45], [46]. The traffic classification community has seen the development from lightweight MLP [47] to more complex RNN [5], [48] and CNN models [21], [49]. Recently, Transformer [17], [18], [19] and Mamba [46], [38] models have been introduced in traffic classification. In this paper, we mainly select CNNs and Transformers as the baseline models for their complexity and high accuracy performance.

B. Lightweight method of neural network.

There exist several lightweight technologies for deep neural networks (DNNs), like quantization [50], [51] and NN pruning [52], [53]. Neural network (NN) quantization reduces the numerical precision of weights and activations to compress models and accelerate inference, while NN pruning removes redundant parameters to compress models while preserving accuracy via iterative retraining. We claim that quantization and pruning are orthogonal to the differential architecture search and dynamic payload distillation in UTFormer.

X. Conclusion

UTFormer is an ultra-lightweight traffic classification model that integrates dynamic payload distillation and differential architecture search, achieving 87.79-99.99% accuracy while reducing computational costs by 9.0-36398.5× and model sizes by 44.9-5433.9× compared to state-of-the-art Transformers. UTFormer eliminates up to 95% of input redundancy with a minimum 106 KB model. Under real-world deployment environments, UTFormer yields 10 Gbps line-rate throughput with a real-time latency of less than 0.2 milliseconds. This work proves that ultra-lightweight UTFormer can rival deep architectures in accuracy while enabling practical improvement in inference throughput performance, paving the way for efficient traffic classification on resource-constrained environments.

References

[1] C. Zheng, Z. Xiong, T. T. Bui, S. Kaupmees, R. Bensoussane, A. Bernabeu, S. Vargaftik, Y. Ben-Itzhak, and N. Zilberman, "IIsy: Hybrid in-network classification using programmable switches," *IEEE/ACM Transactions on Networking*, vol. 32, no. 3, pp. 2555–2570, 2024.

[2] C. Busse-Grawitz, R. Meier, A. Dietmüller, T. Bühler, and L. Vanbever, "pforest: In-network inference with random forests," *arXiv preprint arXiv:1909.05680*, 2019.

[3] G. Xie, Q. Li, G. Duan, J. Lin, Y. Dong, Y. Jiang, D. Zhao, and Y. Yang, "Empowering in-network classification in programmable switches by binary decision tree and knowledge distillation," *IEEE/ACM Transactions on Networking*, vol. 32, no. 1, pp. 382–395, 2023.

[4] L. Yang, S. Fu, Y. Wang, L. Liu, and Y. Luo, "The analysis of encrypted video stream based on low-dimensional embedding method," *IEEE Transactions on Information Forensics and Security, Early Access*, 2024, 10.1109/TIFS.2024.3515816.

[5] K. Kim, J.-H. Lee, H.-K. Lim, S. W. Oh, and Y.-H. Han, "Deep rnn-based network traffic classification scheme in edge computing system," *Computer Science and Information Systems*, vol. 19, no. 1, pp. 165–184, 2022.

[6] J. Yan, H. Xu, Z. Liu, Q. Li, K. Xu, M. Xu, and J. Wu, "Brain-on-switch: Towards advanced intelligent network data plane via nn-driven traffic analysis at line-speed," *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pp. 419–440, 2024.

[7] X. Wang, Y. Wang, Y. Lai, Z. Hao, and A. X. Liu, "Reliable open-set network traffic classification," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 2313–2328, 2025.

[8] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[9] S. Zhao, S. Chen, Y. Sun, Z. Cai, and J. Su, "Identifying known and unknown mobile application traffic using a multilevel classifier," *Security and Communication Networks*, vol. 2019, no. 1, pp. 9 595 081–9 592 099, 2019.

[10] C. Wu, J. Sun, J. Chen, M. Alazab, Y. Liu, and Y. Xiang, "TCG-IDS: Robust network intrusion detection via temporal contrastive graph learning," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 1475–1486, 2025.

[11] J. Ghadermazi, S. Hore, A. Shah, and N. D. Bastian, "GTAE-IDS: Graph transformer-based autoencoder framework for real-time network intrusion detection," *IEEE Transactions on Information Forensics and Security, Early Access*, 2025, 10.1109/TIFS.2025.3557741.

[12] T. Shapira and Y. Shavitt, "FlowPic: A generic representation for encrypted traffic classification and applications identification," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1218–1232, 2021.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4171–4186, 2019.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[15] L. Yang, Y. Wang, L. Liu, J. Huang, J. Shi, S. Fu, and S. Guo, "unFlowS: An unsupervised construction scheme of flow spectrum for network traffic detection," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 3330–3345, 2025.

[16] R. Zhao, X. Deng, Z. Yan, J. Ma, Z. Xue, and Y. Wang, "Mt-flowformer: A semi-supervised flow transformer for encrypted traffic classification," *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2576–2584, 2022.

[17] G. Zhou, X. Guo, Z. Liu, T. Li, Q. Li, and K. Xu, "Trafficformer: an efficient pre-trained model for traffic data," *2025 IEEE Symposium on Security and Privacy (SP)*, pp. 1844–1860, 2024.

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

[19] 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 classification," *Proceedings of the ACM Web Conference 2022*, pp. 633–642, 2022.
[20] D. Wen, T. Li, C. Li, P. Xia, H. Yang, and Z. Sun, "Octopus: A heterogeneous in-network computing accelerator enabling deep learning for network," *arXiv preprint arXiv:2308.11312*, 2023.
[21] M. Gallo, A. Finamore, G. Simon, and D. Rossi, "FENXI: Deep-learning traffic analytics at the edge," *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 202–213, 2021.
[22] W. Wang and L. David, "USTC-TFC dataset." [Online]. Available: https://github.com/yungshenglu/USTC-TFC2016
[23] H.-K. Lim, J.-B. Kim, J.-S. Heo, K. Kim, Y.-G. Hong, and Y.-H. Han, "Packet-based network traffic classification using deep learning," *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pp. 46–51, 2019.
[24] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 43–48, 2017.
[25] W. Zheng, J. Zhong, Q. Zhang, and G. Zhao, "MTT: An efficient model for encrypted network traffic classification using multi-task transformer," *Applied Intelligence*, vol. 52, no. 9, pp. 10 741–10 756, 2022.
[26] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowledge-based Systems*, vol. 212, pp. 106 622–106 643, 2021.
[27] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.
[28] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
[29] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10 012–10 022, 2021.
[30] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," *International Conference on Machine Learning*, pp. 10 347–10 357, 2021.
[31] S. U. Jafri, S. Rao, V. Shrivastav, and M. Tawarmalani, "Leo: Online ml-based traffic classification at multi-terabit line rate," *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pp. 1573–1591, 2024.
[32] Z. Xiong and N. Zilberman, "Do switches dream of machine learning? toward in-network classification," *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, pp. 25–33, 2019.
[33] K. Wang, J. Gao, and X. Lei, "MTC: A multi-task model for encrypted network traffic classification based on transformer and 1d-cnn." *Intelligent Automation & Soft Computing*, vol. 37, no. 1, pp. 619–631, 2023.
[34] ISCX-VPN dataset: https://www.unb.ca/cic/datasets/vpn.html.
[35] T. Van Ede, R. Bortolameotti, A. Continella, J. Ren, D. J. Dubois, M. Lindorfer, D. Choffnes, M. Van Steen, and A. Peter, "Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," *Network and Distributed System Security Symposium (NDSS)*, vol. 27, 2020.
[36] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1928–1943, 2018.
[37] Y. Zhao, G. Dettori, M. Boffa, L. Vassio, and M. Mellia, "The sweet danger of sugar: debunking representation learning for encrypted traffic classification," *Proceedings of the ACM SIGCOMM 2025 Conference*, pp. 296–310, 2025.
[38] T. Wang, X. Xie, W. Wang, C. Wang, J. Liu, B. Huang, Y. Hu, Y. Zhao, and Y. Cui, "NetMamba+: A framework of pre-trained models for efficient and accurate network traffic classification," *arXiv preprint arXiv:2601.21792*, 2026.
[39] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, and M. Matena, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research 21(140)*, pp. 1–67, 2020.
[40] J. Mao, H. Yang, A. Li, H. Li, and Y. Chen, "TPrune: Efficient transformer pruning for mobile devices," *ACM Trans. Cyber-Phys. Syst.*, vol. 5, no. 3, 2021.
[41] A. Zayed, G. Mordido, S. Shabanian, I. Baldini, and S. Chandar, "Fairness-aware structured pruning in transformers," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 20, pp. 22 484–22 492, 2024.
[42] G. Fang, X. Ma, M. Song, M. B. Mi, and X. Wang, "Depgraph: towards any structural pruning," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16 091–16 101, 2023.
[43] M. Dener, S. Al, and G. Ok, "Rfse-gru: Data balanced classification model for mobile encrypted traffic in big data environment," *IEEE Access*, vol. 11, pp. 21 831–21 847, 2023.
[44] P. P. Kundu, T. Truong-Huu, L. Chen, L. Zhou, and S. G. Teo, "Detection and classification of botnet traffic using deep learning with model explanation," *IEEE Transactions on Dependable and Secure Computing*, 2022.
[45] R. Zhao, M. Zhan, X. Deng, Y. Wang, Y. Wang, G. Gui, and Z. Xue, "Yet another traffic classifier: A masked autoencoder based traffic transformer with multi-level flow representation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, pp. 5420–5427, 2023.
[46] T. Wang, X. Xie, W. Wang, C. Wang, Y. Zhao, and Y. Cui, "Netmamba: Efficient network traffic classification via pre-training unidirectional mamba," *2024 IEEE 32nd International Conference on Network Protocols (ICNP)*, pp. 1–11, 2024.
[47] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 258–263, 2016.
[48] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE access*, vol. 5, pp. 18 042–18 050, 2017.
[49] Q. Liao, T. Li, and W. Zhang, "An online network traffic classification method based on deep learning," *2019 IEEE 2nd International Conference on Electronic Information and Communication Technology (ICEICT)*, pp. 34–39, 2019.
[50] S. I. Young, W. Zhe, D. Taubman, and B. Girod, "Transform quantization for cnn compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5700–5714, 2021.
[51] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, "Awq: Activation-aware weight quantization for on-device llm compression and acceleration," *Proceedings of Machine Learning and Systems*, vol. 6, pp. 87–100, 2024.
[52] M. Lin, Y. Zhang, Y. Li, B. Chen, F. Chao, M. Wang, S. Li, Y. Tian, and R. Ji, "1xn pattern for pruning convolutional neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 3999–4008, 2022.
[53] Z. Wang, C. Li, and X. Wang, "Convolutional neural network pruning with structural redundancy reduction," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14 913–14 922, 2021.

Dong Wen is a PhD candidate at National University of Defense Technology. He received the Master degree of computer science at National University of Defense Technology in 2022, and received the Bachelor degree of computer science at Northeastern University in 2019. His research interests includes AI-for-networking, computer architecture and edge computing. Currently he is studying with Prof. Tao Li and Prof. Zhigang Sun.

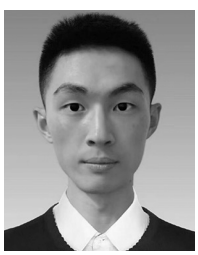
Tianyun Li received the Bachelor of Science degree at the College of Software Engineering , Dalian University of Technology, Dalian, Liaoning Province, and is a Master's student at the College of Computer Science, National University of Defense Technology, Changsha, Hunan Province. Her main research interests are high performance network, data center network, RDMA congestion control and Artificial Intelligence.

Zhuochen Fan received his Ph.D. degree in computer science from Peking University in 2023, advised by Professor Tong Yang. He is currently an Associate Researcher with Pengcheng Laboratory. His research interests include approximation algorithms in security, computer networks, and databases. He had articles published by IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, THE VLDB JOURNAL, ICDE, RTSS, ICPP, ICNP, etc.

Qing Li received the B.S. degree (2008) from Dalian University of Technology, Dalian, China, the Ph.D. degree (2013) from Tsinghua University, Beijing, China; both in computer science and technology. He is currently a full professor at Peng Cheng Laboratory, Shenzhen, China. His research interests include reliable and scalable routing of the Internet, software defined networking, network function virtualization, in-network caching/computing, edge computing, etc.

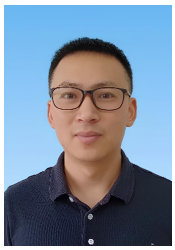
Jie Li , born in 1992. A graduate student at the National University of Defense Technology .His main research interests include traffic classification, deep learning.

Fa Zhu received his Ph.D. in Control Science and Engineering from the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, P.R. China in 2019. From 2016 to 2018, he was a visiting student at the Center for Artificial Intelligence (CAI) and the Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. Now he is an associate professor at the College of Information Science and Technology & College of Artificial Intelligence, Nanjing Forestry University, Nanjing, P.R. China. His current research interests include artificial intelligence, pattern recognition, machine learning and Internet of Things. Most of his researches have published on high prestigious journals, such as IEEE TNNLS, IEEE TIFS, IEEE TSC, IEEE TITS, IEEE TVT, IEEE IV Mag, IEEE TCE, IEEE IOTJ, IEEE JBHI etc. He serves/served as associated editor or guest editor of several journals.



Chenglong Li , born in 1995, PhD, research associate of Defense Innovation Institute, Academy of Military Sciences, Beijing. His main research interests include time-sensitive networking, in-network computing and RISC-V.

Athanasios V. Vasilakos is with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China; and the Center for AI Research, University of Agder, Grimstad, Norway. His main research interests include green networks, sensor nets, the IoTs/mobile nets, smart cities, cloud computing, artificial intelligence/machine learning, block chain technologies, cybersecurity, and big data analytics. He served or is serving as an Editor for many technical journals, such as the IEEE Transactions on Network and Service Management, IEEE Transactions on Cloud Computing, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Cybernetics, IEEE Transactions on Nanobioscience, IEEE Transactions on Information Technology in Biomedicine, ACM Transactions on Autonomous and Adaptive Systems, and the IEEE Journal on Selected Areas in Communications. He is chairing (as general chair) and organizing several international conferences including IEEE ICDCS 2012 1rst Workshop on Forensics, Security and Privacy, ACM BodyNets 2010 ,2011 IEEE INFOCOM 2011 1rst Workshop on Cloud Computing, IEEE SECON 2016- 1rst CoWPER - Toward A City-Wide Pervasive EnviRonment. He is also member of the Technical Program Committee of several conferences. He obtained best paper award of IEEE ICCCN 2014-International Workshop on Hot Topics in Big Data and Networking (HotData I), ACM Mobiquitous 2013, In Proc. of the 5th International Conference on Bio-Inspired Models of Network, Information and Computing Systems (ACM BIONETICS 2010).



Tao Li is the associate professor of College of Computer, National University of Defense Technology. His research focuses on high-performance network chips and systems. He has been in charge of the development of five specialized network chips. His research achievements have been awarded one Second Prize of National Science and Technology Progress, three First Prizes at the provincial and ministerial level for science and technology progress, and one Third Prize for science and technology progress.