Pruned Traffic Trees: Native Semantic Compression with a Protocol-Structured Model Family for Encrypted Traffic Classification

Yuantu Luo*, Jun Tao*, Xiangyu Xu†, Linxiao Yu*, Kangying Li*
* *School of Cyber Science and Engineering, Southeast University*, Nanjing, China
{ytluo, juntao, yulinxiaoybbb, kangying}@seu.edu.cn
† *School of Computer Science and Engineering, Southeast University*, Nanjing, China
xy-xu@seu.edu.cn

***Abstract*—Deep learning has achieved strong performance in encrypted traffic classification (ETC), yet its computational cost limits deployment on resource-constrained network devices such as routers and middleboxes. Existing compression methods mainly operate on weights, channels, hidden representations, or predictions, but do not explicitly determine which protocol fields and structural contexts should remain. We propose Pruned Traffic Trees (PTT), a three-level protocol-structured model family that treats native protocol structures as compression units. PTT-Full learns protocol-structured representations and field salience from complete Protocol Tree Graphs (PTGs), with flow-level self-supervised learning and protocol-presence-aware sparse execution. The learned salience and TopK+k closure construct Distilled PTGs (PTG-Ds) for PTT-Distilled, while PTT-Lite inherits this topology and reduces width through structure-aligned transfer and flow-level logits distillation. Under flow-disjoint and Strong Information Information (SII)-masked settings, PTT-Full achieves Macro-F1 scores of 0.9519 and 0.9416 on CSTNET-TLS1.3 and CipherSpectrum, while PTT-Lite retains 0.9325 and 0.9136 with 80.3% and 61.3% fewer parameters, 98.85% and 98.78% lower effective GFLOPs, and 8.75× and 8.46× CPU inference speedups. These results demonstrate that treating protocol structure itself as the compression object enables effective performance-efficiency trade-offs for lightweight ETC.**

***Index Terms*—Encrypted Traffic Classification, Protocol-Structured Representation, Semantic Compression, Sparse Expert Model**

I. Introduction

Encrypted payloads obscure application content, forcing encrypted traffic classification (ETC) models to rely on the structural evidence that remains observable at network devices [1]. Packet dissection exposes protocol fields, their boundaries, and their positions in the protocol hierarchy, which are interpreted jointly rather than as isolated bytes during traffic analysis. Together, these properties make protocol fields natural semantic units for ETC: they support structure-aware representation and, unlike anonymous bytes or latent channels, can be explicitly retained or removed during compression. A field's meaning depends on both its value and structural position, making field identity and protocol context inseparable. This suggests that ETC models should preserve protocol fields as persistent units throughout both representation learning and compression.

Existing ETC methods commonly encode traffic using packet-direction and timing sequences [2]–[5], raw-byte or token sequences [6], [7], or image-like representations [8], [9]. These representations can achieve strong classification performance, but they do not explicitly preserve protocol-defined field boundaries and parent–child relations. Fixed-length processing may further introduce padding or truncation, which inserts artificial values or removes useful traffic information. Generic graph-based methods improve structural modeling, but their edges are often constructed from statistical or heuristic relations rather than the native hierarchy produced by packet dissection [10]. Consequently, protocol fields are rarely exposed as persistent, semantically identifiable units that can later be selected or removed during model compression.

Beyond representation accuracy, practical ETC deployment should satisfy the latency, memory, and computation budgets of routers and middleboxes. Existing lightweight designs typically compress models through pruning, quantization, width reduction, or knowledge distillation [11]. These methods operate mainly on weights, channels, hidden representations, or predictions, and therefore do not explicitly control which protocol evidence remains available to a compact classifier. Applying structure pruning directly to a graph does not fully resolve this issue either: independently retained nodes may become detached from the protocol paths that define their context. This suggests that lightweight ETC requires more than a smaller neural network. The representation itself should expose semantically meaningful units that can be removed while retaining the protocol context required by the remaining fields. In other words, compression should answer both **what protocol evidence to retain** and **how to retain it in a valid structural context**.

To address these challenges, we present **Pruned Traffic Trees (PTT)**, a protocol-structured ETC model family built on Protocol Tree Graphs (PTGs) [12]. PTGs align parsed protocol fields and their parent–child relations with persistent graph nodes and edges. While the inherited PTG representation uses this structure for representation learning, PTT further turns its field-level alignment into an explicit compression interface. PTT-Full learns protocol-structured representations together

with field salience from complete PTGs. The learned salience determines which fields are retained, while TopK+k closure restores their required protocol paths to construct compact Distilled PTGs (PTG-Ds) and instantiate PTT-Distilled. PTT-Lite then inherits the compressed topology and reduces width through structure-aligned transfer and flow-level logits distillation. To ensure that the measured gains reflect transferable traffic patterns rather than shortcut indicators or flow leakage, all methods are evaluated under SII-masked [13] and flow-disjoint settings [14].

Different from model-level compression that operates on anonymous parameters or latent channels, PTT makes protocol fields and their native hierarchy explicit objects of compression. Overall, our contributions are summarized as follows:

- We propose **PTG-native structural compression**, which turns field-aligned PTGs into an explicit compression interface. Learned field salience determines which protocol fields are retained, while TopK+k closure constructs the minimal hierarchy-closed PTG containing those fields. This reduces graph structure without breaking the native protocol context of the retained evidence.
- We develop **PTT-Full** as the high-capacity source model. It combines flow-level self-supervised learning with protocol-presence-aware sparse dispatch, executing experts only for present protocol components and converting protocol absence into inference acceleration.
- We derive **PTT-Lite** from PTT-Distilled through structure-aligned width transfer and flow-level logits distillation. Experiments on two TLS 1.3 datasets under flow-disjoint and SII-masked settings show that the PTT variants provide strong performance–resource trade-offs.

II. Related Work

a) Encrypted Traffic Representation and Classification: Encrypted traffic classifiers encode flows as images [8], [9], raw-byte or token sequences [3], [4], [15], packet-length and timing sequences [2], [7], or multi-level features [5], [16]. Recent sequence models improve byte-level learning through joint byte–label attention or large-scale traffic pretraining [17]. Graph methods capture packet-, message-, or feature-level interactions [18], while PTGAMoE aligns graph nodes and edges with parsed fields and their native hierarchy [12]. However, these representations either flatten protocol evidence into implicit features or model generic interactions without exposing protocol fields as explicit structural units for compression. Moreover, shortcut features and improper splitting can substantially inflate ETC results [13], [14].

b) Knowledge Distillation and Lightweight Traffic Analysis: Existing lightweight traffic analysis mainly follows general model compression paradigms, including knowledge distillation, pruning, and representation reduction. Graph distillation transfers logits [19]–[21], embeddings [22]–[24], cross-layer relations [25], or neighborhood information to compact students [26], [27]. Other lightweight traffic studies explore programmable-switch deployment, sample-efficient learning, graph cross-distillation, and contextual masking distillation [28]–[31]. However, these methods mainly compress model parameters or representations rather than explicitly preserving the protocol context of retained fields. Moreover, they neither select the protocol evidence to retain nor preserve its native hierarchy.

c) Mixture of Experts for Traffic Analysis: Sparse Mixture-of-Experts (MoE) models improve capacity–computation trade-offs through conditional expert activation [32], [33]. Recent traffic models use experts for sparse foundation modeling, heterogeneous traffic specialization, and multi-view fusion [34]–[36]. However, their routing decisions are typically defined over learned tokens, modalities, or predefined branches, rather than protocol components physically present in each packet.

PTT differs by treating parsed fields and their hierarchy as explicit semantic units for both sparse execution and compression. It leverages protocol presence as an explicit routing signal and further uses learned field salience to guide hierarchy-preserving protocol compression.

III. Protocol-Structured Source Model PTT-Full

As shown in Fig. 1, building on the Protocol Tree Graph (PTG) representation [12] and graph-expert backbone, PTT-Full adds flow-level self-supervised learning and protocol-presence-aware sparse dispatch.

A. Graph-Based Expert Backbone

Let $\mathcal{D} = \{(F_i, y_i)\}_{i=1}^{N}$ be a labeled ETC dataset, where $F_i = (p_{i,1}, \ldots, p_{i,T_i})$ is a flow and y_i is its class. For a maximum of P packets per flow, $T_i = \min(N_i, P)$ retains the first packets in capture order.

Each packet is represented over an expert-indexed PTG set $\{\mathcal{G}_{F,e} \mid e \in \mathcal{E}\}$. For expert e, the fixed Full schema is

$$\mathcal{G}_{F,e} = (\mathcal{V}_{F,e}, \mathcal{E}_{F,e}), \tag{1}$$

$$\mathcal{V}_{F,e} = \mathcal{V}_{F,e}^{\mathrm{real}} \cup \mathcal{V}_{F,e}^{\mathrm{abs}}, \tag{2}$$

$$\mathcal{E}_{F,e} = \mathcal{E}_{F,e}^{\mathrm{hier}} \cup \mathcal{E}_{F,e}^{\mathrm{sink}}. \tag{3}$$

Here, $\mathcal{V}_{F,e}^{\mathrm{real}}$ contains real protocol-field nodes whose values are obtained from packet dissection, whereas $\mathcal{V}_{F,e}^{\mathrm{abs}}$ contains abstract schema nodes used to preserve the protocol hierarchy. $\mathcal{E}_{F,e}^{\mathrm{hier}}$ contains undirected hierarchical edges induced by the parent–child relations of packet dissection, while $\mathcal{E}_{F,e}^{\mathrm{sink}}$ connects schema nodes with the layer-wise sink node used for graph readout. Each PTG is conceptually undirected. In implementation, every undirected edge is represented by two directed edges for bidirectional GAT message passing.

For packet $p_{i,t}$, $\mathbf{X}_{i,t,e} \in \mathbb{R}^{|\mathcal{V}_{F,e}| \times d_F}$ contains node representations in the fixed schema order. Each node v has a shared gate logit $\alpha_{e,v}$ and activation

$$s_{e,v} = \sigma(\alpha_{e,v}), \qquad 0 < s_{e,v} < 1. \tag{4}$$

The gated node representations and expert output are

$$\widetilde{\mathbf{X}}_{i,t,e} = \left[s_{e,v}\, \mathrm{LN}(\mathbf{x}_{i,t,e,v}) \right]_{v \in \mathcal{V}_{F,e}},$$

$$\mathbf{h}_{i,t,e} = f_e(\widetilde{\mathbf{X}}_{i,t,e}, \mathcal{E}_{F,e}; \theta_{F,e}), \qquad \mathbf{h}_{i,t,e} \in \mathbb{R}^{d_F}. \tag{5}$$

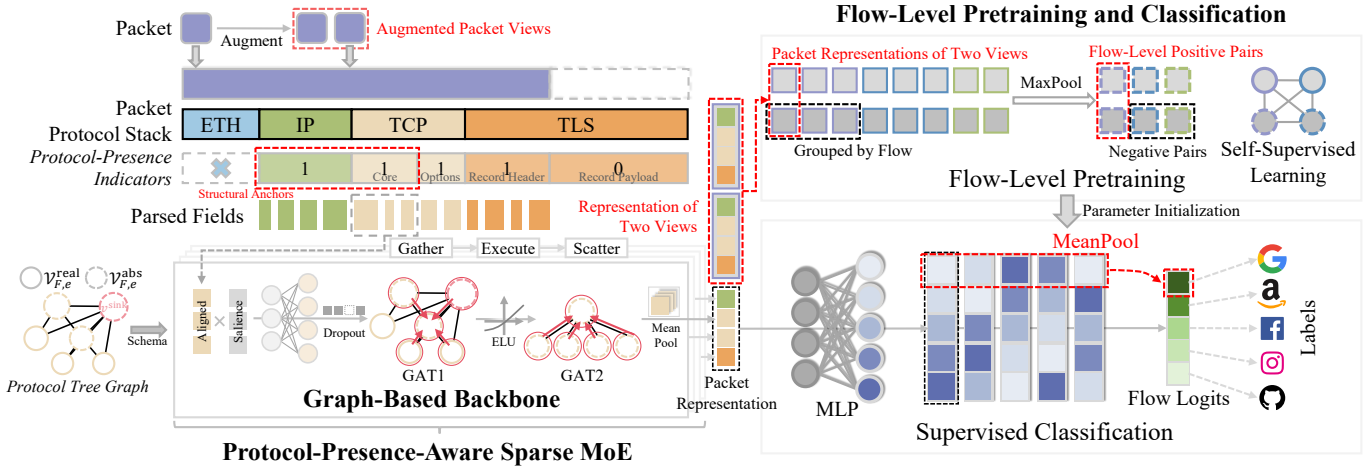


Fig. 1: Overview of PTT-Full and its flow-level learning pipeline.

Here, f_e is a two-layer graph attention encoder followed by mean readout:

$$\begin{aligned}
\mathbf{Z}^{(0)}_{i,t,e} &= \mathrm{Dropout}(\widetilde{\mathbf{X}}_{i,t,e}),\\
\mathbf{Z}^{(1)}_{i,t,e} &= \mathrm{ELU}\left(\mathrm{GAT}^{(1)}_e(\mathbf{Z}^{(0)}_{i,t,e}, \mathcal{E}_{F,e})\right),\\
\mathbf{Z}^{(2)}_{i,t,e} &= \mathrm{GAT}^{(2)}_e(\mathbf{Z}^{(1)}_{i,t,e}, \mathcal{E}_{F,e}),\\
\mathbf{h}_{i,t,e} &= \mathrm{MeanPool}(\mathbf{Z}^{(2)}_{i,t,e}).
\end{aligned} \tag{6}$$

The first layer uses multi-head attention and concatenation, while the second outputs dimension d_F. The learned real-node gates provide field salience for structural compression.

Beyond the PTG schema and its corresponding graph-expert architecture, we introduce flow batching with variable lengths, flow-level self-supervised learning, and protocol-presence-aware sparse execution in PTT-Full to enhance the traffic representation.

B. Protocol-Presence-Aware Sparse MoE

a) Flow-Centric Batching: To train and evaluate PTT at the flow granularity, packets from the same flow are first organized into a variable-length micro-batch. The packet indices of flow i form the micro-batch

$$\mathcal{I}^{\mathrm{pkt}}_i = \{(i,t) \mid 1 \le t \le T_i\}. \tag{7}$$

For each optimization step, the sampler selects up to $N_{\mathcal{B}}$ flows with index set $\mathcal{I}^{\mathrm{flow}}_{\mathcal{B}}$ and concatenates their variable-length packet sets into a macro-batch:

$$\mathcal{I}^{\mathrm{pkt}}_{\mathcal{B}} = \biguplus_{i \in \mathcal{I}^{\mathrm{flow}}_{\mathcal{B}}} \mathcal{I}^{\mathrm{pkt}}_i, \qquad |\mathcal{I}^{\mathrm{pkt}}_{\mathcal{B}}| \le N_{\mathcal{B}} P. \tag{8}$$

This macro-batch introduces no packet padding, and flow identifiers are retained to regroup packet outputs for flow-level learning.

b) Protocol Presence: A packet should not invoke an expert for a protocol component that it does not contain. For packet $p_{i,t}$ and expert e, define

$$\begin{aligned}
\mathcal{P}_{i,t,e} &= \{v \in \mathcal{V}^{\mathrm{real}}_{F,e} \mid v \text{ is observed in } p_{i,t}\},\\
m_{i,t,e} &= \mathbb{I}[\mathcal{P}_{i,t,e} \neq \varnothing].
\end{aligned} \tag{9}$$

For the TCP-based flows considered here, IP and TCP-core experts are retained as structural anchors:

$$\begin{aligned}
\mathcal{E}_{\mathrm{anchor}} &= \{e_{\mathrm{IP}}, e_{\mathrm{TCPcore}}\},\\
m'_{i,t,e} &= \max(m_{i,t,e}, \mathbb{I}[e \in \mathcal{E}_{\mathrm{anchor}}]).
\end{aligned} \tag{10}$$

where $\mathbb{I}[\cdot]$ denotes the indicator function. The active packet indices of expert e are gathered into an expert-specific sub-batch

$$\mathcal{I}^{\mathrm{act}}_{\mathcal{B},e} = \{(i,t) \in \mathcal{I}^{\mathrm{pkt}}_{\mathcal{B}} \mid m'_{i,t,e} = 1\}. \tag{11}$$

Thus, protocol presence determines whether an expert can execute, while the fusion gate below determines how much its output contributes.

c) Sparse Dispatch and Fusion: Masking an expert only after its forward pass does not reduce computation. PTT-Full therefore uses a gather–execute–scatter procedure: active packets are gathered into expert-specific sub-batches, each nonempty encoder executes once, and its outputs are scattered back to the original packet positions. In particular,

$$\mathcal{I}^{\mathrm{act}}_{\mathcal{B},e} = \varnothing \quad \Longrightarrow \quad f_e \text{ is not executed.} \tag{12}$$

Inactive positions are zero-filled:

$$\overline{\mathbf{h}}_{i,t,e} = \begin{cases} \mathbf{h}_{i,t,e}, & m'_{i,t,e} = 1,\\ \mathbf{0}, & m'_{i,t,e} = 0. \end{cases} \tag{13}$$

Given expert order $(e_1, \ldots, e_{|\mathcal{E}|})$, cooperative fusion is

$$\begin{aligned}
\boldsymbol{\beta}_{i,t} &= \mathrm{MLP}_g(\overline{\mathbf{h}}_{i,t,e_1} \| \cdots \| \overline{\mathbf{h}}_{i,t,e_{|\mathcal{E}|}}),\\
g_{i,t,e} &= m'_{i,t,e}\sigma(\beta_{i,t,e}),\\
\mathbf{z}_{i,t} &= \sum_{e \in \mathcal{E}} g_{i,t,e}\overline{\mathbf{h}}_{i,t,e}.
\end{aligned} \tag{14}$$

Unlike competitive softmax routing, sigmoid fusion allows coexisting components such as IP, TCP, and TLS to contribute simultaneously.

C. Flow-Level Pretraining and Classification

Packets from the same flow describe one communication process. PTT-Full therefore defines contrastive relations at the flow level rather than treating packets as independent samples. During pretraining, two views perturb node observations and selected auxiliary sink edges while sharing the canonical PTGs, expert assignments, and presence masks. These temporary perturbations do not change the schemas later used for compression.

The sparse backbone produces packet representations $\mathbf{z}_{i,t}^{(1)}$ and $\mathbf{z}_{i,t}^{(2)}$. They are pooled within each flow:

$$\mathbf{r}_i^{(\xi)} = \underset{(i,t)\in\mathcal{I}_i^{\mathrm{pkt}}}{\mathrm{MaxPool}}\, \mathbf{z}_{i,t}^{(\xi)}, \qquad \xi \in \{1,2\},$$
$$\mathcal{L}_{\mathrm{SSL}} = \mathrm{NTXent}\left(\{\mathbf{r}_i^{(1)}, \mathbf{r}_i^{(2)} \mid i \in \mathcal{I}_{\mathcal{B}}^{\mathrm{flow}}\}\right). \tag{15}$$

Here, NTXent denotes the normalized temperature-scaled cross-entropy (NT-Xent) loss. The two views of the same flow form a positive pair, while representations from other flows in the macro-batch form negatives. Packets from the same flow are never treated as negatives.

For supervised adaptation, the pretrained backbone is jointly fine-tuned with a classifier:

$$\boldsymbol{\ell}_{i,t} = \mathrm{MLP}_c(\mathbf{z}_{i,t}), \qquad \boldsymbol{\ell}_{i,t} \in \mathbb{R}^C. \tag{16}$$

Packet logits are averaged within each flow:

$$\overline{\boldsymbol{\ell}}_i = \frac{1}{T_i} \sum_{(i,t)\in\mathcal{I}_i^{\mathrm{pkt}}} \boldsymbol{\ell}_{i,t}. \tag{17}$$

The supervised objective is

$$\mathcal{L}_{\mathrm{Full}}^{(\mathrm{sup})} = \mathbb{E}_{(F_i,y_i)\sim\widehat{\mathcal{D}}_{\mathrm{tr}}} \left[\mathrm{CE}(\overline{\boldsymbol{\ell}}_i, y_i)\right], \tag{18}$$

where CE denotes the cross-entropy loss, and $\widehat{y}_i = \arg\max_c \overline{\ell}_{i,c}$ at inference.

Let θ_F^* be the optimized parameters and

$$\mathcal{G}_F = \{\mathcal{G}_{F,e} \mid e \in \mathcal{E}\},$$
$$M_F = (\theta_F^*, \mathcal{G}_F, d_F). \tag{19}$$

The trained PTT-Full serves as both the high-capacity classifier and the source of field salience, PTG structure, and parameters for compression.

IV. PTG-Native Structural Compression

Sparse dispatch skips absent experts, but each active expert still processes a complete PTG. As shown in Fig. 2, PTT-Distilled compresses these PTGs by selecting salient fields and applying TopK+k closure to preserve their structural context, while PTT-Lite inherits the compressed topology and further reduces representation width.

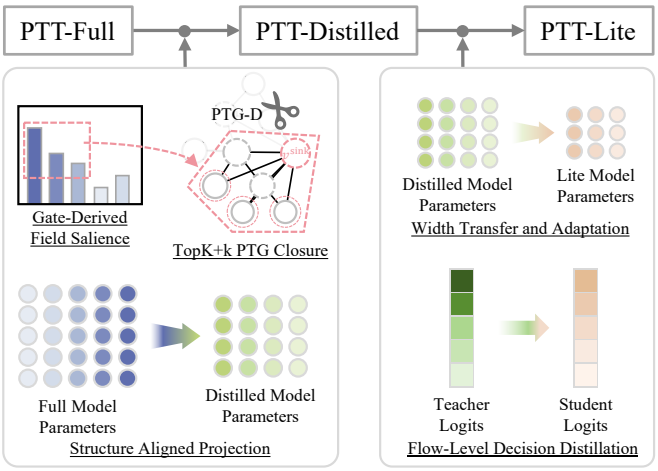


Fig. 2: PTG-native structural compression from PTT-Full to PTT-Distilled and PTT-Lite.

A. PTT-Full to PTT-Distilled

a) Gate-Derived Field Salience: The gates learned by PTT-Full provide a shared salience score for each real protocol field. For expert e, the candidate set is

$$\mathcal{C}_e = \mathcal{V}_{F,e}^{\mathrm{real}}. \tag{20}$$

We normalize the trained gates within each expert to obtain Normalized Field Salience (NFS):

$$\mathrm{NFS}_e(v) = \frac{s_{e,v}^*}{\sum_{u\in\mathcal{C}_e} s_{e,u}^*}, \qquad v \in \mathcal{C}_e. \tag{21}$$

NFS expresses the learned gate values on a relative scale within each expert and is used only for field ranking. Unlike causal attribution methods, NFS is designed as a compression-oriented criterion. Its validity is evaluated by whether the resulting protocol-closed subgraphs preserve classification performance under matched compression budgets. Given a per-expert budget K, we construct the Top-K salience set as:

$$K_e = \min(K, |\mathcal{C}_e|),$$
$$\mathcal{S}_e^K = \mathrm{TopK}_{v\in\mathcal{C}_e}(\mathrm{NFS}_e(v), K_e). \tag{22}$$

b) TopK+k PTG Closure: Retaining the Top-K fields answers what evidence should be preserved, but not how these fields remain meaningful after compression. We therefore design TopK+k closure to restore the minimal protocol hierarchy required by these fields.

Although the PTG is undirected, its hierarchical edges retain the parent–child provenance inherited from packet dissection. We represent this provenance by $\mathrm{Par}_{F,e}(v)$, the parent set of node v in the original protocol hierarchy. As shown in Algorithm 1, starting from $\mathcal{S}_e^K$, PTT recursively follows these parent relations until all required ancestors are retained. Let $\mathcal{V}_e^{\mathrm{core}}$ denote the selected fields together with the PTG schema nodes recovered by this closure.

The additional closure size is

$$k_e = |\mathcal{V}_e^{\mathrm{core}} \setminus \mathcal{S}_e^K|, \qquad |\mathcal{V}_e^{\mathrm{core}}| = K_e + k_e. \tag{23}$$

Algorithm 1: TopK+k PTG Closure

Input: Full PTGs $\{\mathcal{G}_{F,e}\}_{e\in\mathcal{E}}$; parent maps $\{\mathrm{Par}_{F,e}\}_{e\in\mathcal{E}}$; selected field sets $\{\mathcal{S}_e^K\}_{e\in\mathcal{E}}$

Output: Distilled PTGs $\mathcal{G}_D$; closure sizes $\{k_e\}_{e\in\mathcal{E}}$

$\mathcal{G}_D \leftarrow \varnothing$;
foreach $e \in \mathcal{E}$ **do**
  $\mathcal{V}_e^{\mathrm{core}} \leftarrow \mathcal{S}_e^K$;
  $\mathcal{Q}_e \leftarrow \mathcal{S}_e^K$;
  while $\mathcal{Q}_e \neq \varnothing$ **do**
    select and remove one node v from $\mathcal{Q}_e$;
    foreach $u \in \mathrm{Par}_{F,e}(v)$ **do**
      if $u \notin \mathcal{V}_e^{\mathrm{core}}$ **then**
        $\mathcal{V}_e^{\mathrm{core}} \leftarrow \mathcal{V}_e^{\mathrm{core}} \cup \{u\}$;
        $\mathcal{Q}_e \leftarrow \mathcal{Q}_e \cup \{u\}$;
      end
    end
  end
  $k_e \leftarrow |\mathcal{V}_e^{\mathrm{core}} \setminus \mathcal{S}_e^K|$;
  $\mathcal{V}_{D,e} \leftarrow \mathcal{V}_e^{\mathrm{core}} \cup \{v_{F,e}^{\mathrm{sink}}\}$;
  $\mathcal{E}_{D,e}^{\mathrm{hier}} \leftarrow \{(u,v) \in \mathcal{E}_{F,e}^{\mathrm{hier}} \mid u,v \in \mathcal{V}_e^{\mathrm{core}}\}$;
  $\mathcal{E}_{D,e}^{\mathrm{sink}} \leftarrow \{(v, v_{F,e}^{\mathrm{sink}}) \mid v \in \mathcal{V}_e^{\mathrm{core}}\}$;
  $\mathcal{E}_{D,e} \leftarrow \mathcal{E}_{D,e}^{\mathrm{hier}} \cup \mathcal{E}_{D,e}^{\mathrm{sink}}$;
  $\mathcal{G}_{D,e} \leftarrow (\mathcal{V}_{D,e}, \mathcal{E}_{D,e})$;
  $\mathcal{G}_D \leftarrow \mathcal{G}_D \cup \{\mathcal{G}_{D,e}\}$;
end
return $\mathcal{G}_D, \{k_e\}_{e\in\mathcal{E}}$;

The auxiliary sink is added only to preserve the global aggregation structure of the PTG and does not contribute to k_e, giving $|\mathcal{V}_{D,e}| = K_e + k_e + 1$.

Ignoring the auxiliary sink node, $\mathcal{V}_e^{\mathrm{core}}$ is the unique minimal hierarchy-preserving node set that contains $\mathcal{S}_e^K$ and is closed under the parent relation inherited from the original protocol hierarchy. Any smaller node set would necessarily remove either a selected field or an ancestor required by one of its original protocol paths. The value of k_e is therefore determined by the protocol paths of the selected fields rather than by a fixed pruning ratio.

Since $\mathcal{V}_e^{\mathrm{core}}$ prevents a node from being enqueued more than once, the ancestor traversal visits each discovered node once. Constructing $\mathcal{E}_{D,e}^{\mathrm{hier}}$ requires one scan of the Full hierarchical edge set, giving an overall complexity of $\mathcal{O}(|\mathcal{V}_{F,e}| + |\mathcal{E}_{F,e}^{\mathrm{hier}}|)$ for each expert.

c) Structure-Aligned Projection: The PTG-D topology is fixed before parameter projection. For $d_D < d_F$, the Distilled model is initialized as

$$\begin{aligned} \theta_D^{(0)} &= \Pi_{F\to D}(\theta_F^*; \mathcal{G}_D), \\ M_D^{(0)} &= (\theta_D^{(0)}, \mathcal{G}_D, d_D). \end{aligned} \tag{24}$$

Here, $\Pi_{F\to D}$ denotes structure-aligned channel selection. Specifically, $\Pi_{F\to D}$ selects the top-d_D channels by aggregated squared-ℓ_2 weight magnitude and slices them consistently across aligned model modules, while retained PTG nodes are matched by protocol-field name. The model is then adapted with flow-level supervision:

$$\begin{aligned} \mathcal{L}_{\mathrm{Distilled}}^{(\mathrm{sup})} &= \mathbb{E}_{(F_i,y_i)\sim\widehat{\mathcal{D}}_{\mathrm{tr}}}[\mathrm{CE}(\overline{\boldsymbol{\ell}}_{D,i}, y_i)], \\ \theta_D^* &= \arg\min_{\theta_D} \mathcal{L}_{\mathrm{Distilled}}^{(\mathrm{sup})}, \\ M_D &= (\theta_D^*, \mathcal{G}_D, d_D). \end{aligned} \tag{25}$$

PTT-Distilled retains the expert set, anchor experts, and sparse dispatch rule of PTT-Full, but each active expert processes a smaller graph.

B. PTT-Distilled to PTT-Lite

Unlike the previous stage, this stage removes no protocol field. PTT-Lite keeps $\mathcal{G}_D$ unchanged and only reduces representation width from d_D to $d_L < d_D$.

a) Width Transfer and Adaptation: Because PTT-Distilled and PTT-Lite share the same PTG-D schemas and expert organization, their node-specific parameters are structurally aligned:

$$\begin{aligned} \theta_L^{(0)} &= \Pi_{D\to L}(\theta_D^*; \mathcal{G}_D), \\ M_L^{(0)} &= (\theta_L^{(0)}, \mathcal{G}_D, d_L). \end{aligned} \tag{26}$$

$\Pi_{D\to L}$ applies the same criterion to select the top-d_L channels and slices the corresponding dimensions consistently across all aligned modules. The projected experts are briefly frozen while the narrower fusion and classifier adapt to the new feature space. All parameters are then jointly fine-tuned using

$$\mathcal{L}_{\mathrm{Lite}}^{(\mathrm{sup})} = \mathbb{E}_{(F_i,y_i)\sim\widehat{\mathcal{D}}_{\mathrm{tr}}}[\mathrm{CE}(\overline{\boldsymbol{\ell}}_{L,i}, y_i)]. \tag{27}$$

b) Flow-Level Logits Distillation: Because the Lite model has lower capacity, it is additionally guided by the flow-level predictions of PTT-Distilled. With temperature τ_{KD},

$$q_X(c \mid F_i) = \frac{\exp(\overline{\ell}_{X,i}^{c}/\tau_{\mathrm{KD}})}{\sum_{c'=1}^{C}\exp(\overline{\ell}_{X,i}^{c'}/\tau_{\mathrm{KD}})}, \qquad X \in \{D, L\}. \tag{28}$$

The Distilled teacher is fixed during Lite adaptation, and

$$\mathcal{L}_{\mathrm{KD}} = \tau_{\mathrm{KD}}^2 \mathbb{E}_{F_i\sim\widehat{\mathcal{D}}_{\mathrm{tr}}}\left[\mathrm{KL}(q_D(\cdot \mid F_i) \| q_L(\cdot \mid F_i))\right]. \tag{29}$$

The final objective is

$$\begin{aligned} \mathcal{L}_{\mathrm{Lite}} &= (1-\lambda_{\mathrm{KD}})\mathcal{L}_{\mathrm{Lite}}^{(\mathrm{sup})} + \lambda_{\mathrm{KD}}\mathcal{L}_{\mathrm{KD}}, \\ \theta_L^* &= \arg\min_{\theta_L} \mathcal{L}_{\mathrm{Lite}}, \\ M_L &= (\theta_L^*, \mathcal{G}_D, d_L). \end{aligned} \tag{30}$$

This loss transfers flow-level decisions without changing the PTG-D topology.

The inherited PTG backbone represents each protocol field with a persistent schema node whose identity is shared across packets. PTT further uses this field-level alignment as a compression interface: learned node gates determine which protocol evidence is retained, while the protocol hierarchy determines how the retained evidence forms a valid reduced graph. This differs from compressing anonymous hidden channels or independently removing graph nodes, because every

structural decision remains associated with a concrete protocol field and its original context. The same field identities are preserved across PTT-Full, PTT-Distilled, and PTT-Lite, which also provides a natural alignment for cross-stage parameter transfer.

V. Evaluation

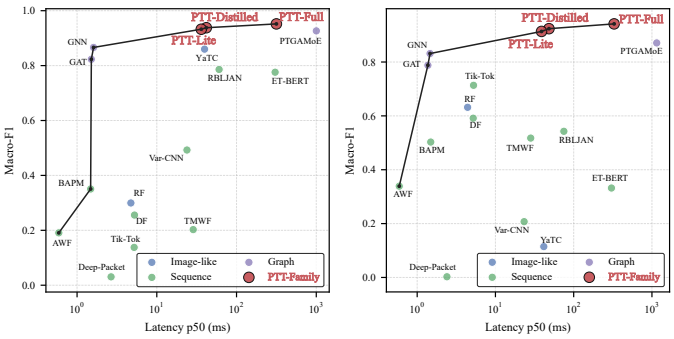
A. Experimental Setup

a) Datasets and Evaluation Protocol: We evaluate PTT on CSTNET-TLS1.3, which contains encrypted sessions across 26 domains, and CipherSpectrum, which contains 120,000 TLS 1.3 sessions across 41 domains and three cipher suites. All methods use identical flow-disjoint splits and SII-masked packet traces. The complete ETH layer, IP addresses, transport ports, and server names are removed before constructing each model's input representation. PTT uses no composite TLS fingerprints such as JA3/JA4. Retained TLS fields remain individual nodes in their protocol hierarchy.

b) Compared Methods: We compare PTT-Full, PTT-Distilled, and PTT-Lite with image-like, byte/sequence, and graph classifiers, including RF [9], YaTC [8], TMWF [7], BAPM [2], Var-CNN [3], DF [4], Tik-Tok [5], AWF [16], Deep-Packet [6], ET-BERT [17], RBLJAN [15], GNN, and GAT. PTGAMoE [12], which uses dense PTG experts and supervised training, serves as the direct architectural baseline. All baselines are retrained under the same flow-disjoint splits and SII-masked inputs, rather than using their originally reported results.

c) Metrics and Inference Measurement: Accuracy and Macro-F1 measure classification performance. We additionally report parameter count (Params), effective GFLOPs (GFLOPs), CPU p50 latency (Lat.), and flow throughput (flow/s) to evaluate efficiency. Unless stated otherwise, GFLOPs count only the experts actually executed. Latency is measured over one macro-batch of $N_{\mathcal{B}} = 32$ flows, each retaining at most $P = 32$ packets. Packet parsing, protocol dissection, and input construction are excluded for all methods. For PTT, the timed forward pass includes gathering, sparse dispatch, graph execution, scattering, expert fusion, and classification. Flow throughput is computed as $\text{Flows/s} = \frac{N_{\mathcal{B}}}{\text{Latency}/1000}$, where latency is measured in milliseconds per macro-batch. All models are benchmarked on the same CPU with the same macro-batch size, packet limit, and timing boundary.

d) Implementation Details: Code is available at https://anonymous.4open.science/r/pruned_traffic_trees-D08C. Training uses a server with Intel i5-13490F CPU, 32 GB memory, and an NVIDIA RTX 4060 Ti 16 GB GPU. CPU inference uses a server with Intel i5-9500T and 8 GB memory. PTT-Full uses IP, TCP-core, TCP-option, TLS-record-header, and TLS-record-payload experts with $d_F = 128$. Unless otherwise stated, PTT-Distilled uses $K = 5$ and $d_D = 32$, and PTT-Lite uses $d_L = 20$. The Lite KD weights are 0.10 on CSTNET-TLS1.3 and 0.05 on CipherSpectrum.



(a) CSTNET-TLS1.3. (b) CipherSpectrum.

Fig. 3: Pareto frontiers of performance–latency trade-offs.

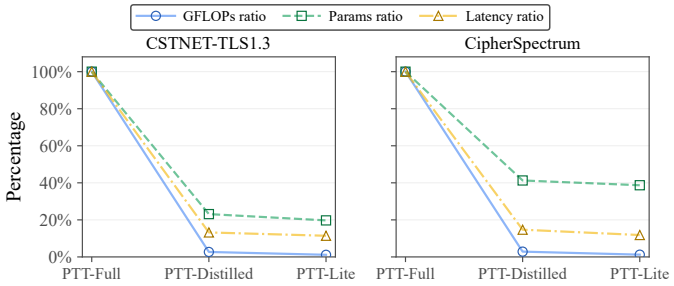


Fig. 4: Compression effectiveness of PTT Family.

B. Overall Performance and CPU Efficiency

Table I shows that PTT-Full achieves the highest Macro-F1 on both datasets. PTT-Lite retains 0.9325 and 0.9136 Macro-F1 with 0.186M and 0.481M parameters, exceeding GAT by 0.0663 and 0.0821, respectively. Compared with dense PTGAMoE, PTT-Full improves Macro-F1 by 0.0254 and 0.0707 while reducing macro-batch latency by 3.17× and 3.50×.

To jointly compare performance and latency, Fig. 3 plots the performance–latency trade-off leveraging Pareto frontiers. A model is non-dominated when no alternative has both higher Macro-F1 and lower latency. The PTT variants occupy the high-performance region on both datasets and provide distinct Full, Distilled, and Lite operating points.

C. Compression Effectiveness

As shown in Fig. 4, the Full-to-Distilled transition, which jointly reduces PTG topology and representation width, accounts for most of the overall resource reduction. Table III further isolates the advantage of PTG-native structural compression under matched computation budgets. PTT-Lite keeps the distilled topology unchanged and further narrows the representation width. It uses only 19.72% and 38.70% of the Full-model parameters and 1.15% and 1.22% of its effective GFLOPs. The corresponding CPU speedups are 8.75× and 8.46×, indicating that structural compression provides the main reduction and width compression adds a lighter deployment point. This confirms that reducing protocol structures

TABLE I: Overall comparison on CSTNET-TLS1.3 and CipherSpectrum.

Dataset		CSTNET-TLS1.3				CipherSpectrum			
Category	Method	AC	F1	Params (M)	Lat. (ms/batch)	AC	F1	Params (M)	Lat. (ms/batch)
Image-like	RF	0.3648	0.2996	0.924	4.73	0.6251	0.6318	0.947	4.41
	YaTC	0.8756	0.8599	1.863	39.72	0.1162	0.1146	1.866	41.49
Byte/Sequence	TMWF	0.2953	0.2026	4.834	28.62	0.5289	0.5175	4.838	28.24
	BAPM	0.4069	0.3505	0.238	1.48	0.5097	0.5028	0.269	1.49
	Var-CNN	0.4963	0.4925	8.766	23.89	0.2455	0.2072	8.782	23.27
	DF	0.3449	0.2555	3.679	5.25	0.5949	0.5913	3.687	5.20
	Tik-Tok	0.2233	0.1378	3.679	5.20	0.7176	0.7135	3.687	5.25
	AWF	0.2655	0.1908	0.048	0.59	0.3640	0.3389	0.069	0.59
	Deep-Packet	0.0656	0.0305	10.108	2.67	0.0563	0.0026	10.109	2.40
	ET-BERT	0.7761	0.7759	132.146	304.81	0.4059	0.3321	132.157	303.88
	RBLJAN	0.8266	0.7859	0.357	60.44	0.5808	0.5427	0.499	74.88
Graph	GNN	0.8530	0.8230	0.026	1.51	0.7888	0.7880	0.029	1.37
	GAT	0.8905	0.8662	0.141	1.61	0.8337	0.8315	0.144	1.46
	PTGAMoE	0.9395	0.9265	0.962	1001.93	0.8720	0.8709	1.263	1150.57
PTT	PTT-Full	**0.9646**	**0.9519**	0.943	316.20	**0.9415**	**0.9416**	1.243	328.66
	PTT-Distilled	0.9472	0.9380	0.218	41.79	0.9241	0.9238	0.513	48.31
	PTT-Lite	0.9424	0.9325	0.186	36.12	0.9137	0.9136	0.481	38.85

rather than only hidden dimensions is the main source of compression efficiency.

D. Effect of Presence-Aware Sparse Modeling

Table II separates the modeling effect of protocol-presence masking from the efficiency gain of actual sparse execution. Adding SSL alone (D1) provides limited gains over D0, whereas presence masking (D2) improves Macro-F1 and further combines with SSL (D3) to reach 0.9519 and 0.9416. D3 and D4 share the same checkpoints and predictions. D4 only enables sparse execution for absent protocol components. It reduces GFLOPs by 24.7% and 48.8%, latency by 27.9% and 48.4%, and increases throughput by 36.6% and 91.6%. Thus, presence masking improves protocol-aware modeling, while sparse execution turns protocol absence into computation savings without changing predictions.

E. Effect of PTG-Native Structural Compression

This section examines four questions: (1) whether PTG-native compression is more effective than model-level compression; (2) whether learned field salience guides selection; (3) whether retained fields require their native hierarchy; and (4) how the field budget controls PTG-D size.

a) Model-Level versus PTG-Native Compression: We first compare PTT with three validation-tuned model-level baselines that retain the complete PTG. Narrow-CE (N-CE) reduces model width and trains the resulting full-PTG model with cross-entropy, while Narrow-KD (N-KD) additionally transfers PTT-Full logits. As a structured pruning baseline, ℓ_2-based channel pruning (L2-Ch) ranks channels by the aggregated ℓ_2 norm of their associated weights and retains the highest-ranked channels before validation-tuned adaptation.

To ensure compute-matched comparison, we enumerate full-PTG widths $d \in \{4, 8, 12, 16, 20, 24, 28, 32\}$ and select the largest width whose validation-set effective GFLOPs do not exceed the corresponding PTT-Distilled or PTT-Lite budget. This yields $d = 20$ and $d = 12$, respectively, on both datasets. All baselines further tune the learning rate on $\{10^{-4}, 2 \times 10^{-4}, 5 \times 10^{-4}\}$, while N-KD and L2-Ch also tune their KD weights using validation data. The checkpoint with the highest validation Macro-F1 is then evaluated on the test set.

Table III shows that PTT consistently retains more classification performance than the strongest validation-tuned model-level baseline. Its Macro-F1 gains are 0.0220 and 0.0324 at the Distilled budget, and increase to 0.0404 and 0.0743 at the Lite budget on CSTNET-TLS1.3 and CipherSpectrum, respectively. Although the full-PTG baselines use slightly fewer effective GFLOPs, their CPU latency is approximately $1.83\times$ higher at the Distilled budget and $1.80\times$ higher at the Lite budget. These results indicate that removing less informative protocol structure provides a better performance–efficiency trade-off than retaining the complete PTG and compressing model width or channels alone.

b) Field Guidance and Salience Stability: We next examine which real protocol fields should be retained before closure. Random (Rand.) samples candidate fields uniformly, Frequency (Freq.) ranks them by observation frequency in the training set, Taylor [37] uses first-order training-loss sensitivity, and NFS ranks fields according to the gates learned by PTT-Full. Because the NFS normalization is shared by all candidates within the same expert, it does not change their ordering. The resulting Top-K selection is determined by the learned gate salience. All methods use the same $K = 5$ and TopK+k closure.

As shown in Fig. 5, the learned gate salience produces the strongest compressed models under comparable graph-computation budgets. We further examine the stability of the selected fields across three independent training runs. For experts with more than five candidate fields, the selected

TABLE II: Effect of SSL, protocol-presence masking, and actual expert skipping.

Dataset	ID	SSL	Mask	Sparse	F1	GFLOPs	Lat. (ms/batch)	Flows/s
CSTNET-TLS1.3	D0	✗	✗	✗	0.9296 (±0.0146)	9.241	436.04 [28.95]	73.54 [4.51]
	D1	✓	✗	✗	0.9301 (±0.0046)	9.241	436.94 [32.34]	73.57 [4.63]
	D2	✗	✓	✗	0.9415 (±0.0042)	9.241	436.87 [28.63]	73.51 [4.52]
	D3	✓	✓	✗	0.9519 (±0.0111)	9.241	438.35 [31.08]	73.40 [4.80]
	D4	✓	✓	✓	**0.9519** (±0.0111)	**6.961**	**316.20** [18.53]	**100.28** [6.00]
CipherSpectrum	D0	✗	✗	✗	0.9322 (±0.0042)	12.444	635.04 [24.82]	50.17 [2.13]
	D1	✓	✗	✗	0.9360 (±0.0074)	12.444	641.06 [26.53]	49.52 [2.04]
	D2	✗	✓	✗	0.9402 (±0.0036)	12.444	639.01 [25.61]	49.74 [2.12]
	D3	✓	✓	✗	0.9416 (±0.0013)	12.444	637.52 [25.44]	49.96 [2.26]
	D4	✓	✓	✓	**0.9416** (±0.0013)	**6.371**	**328.66** [14.82]	**95.73** [4.51]

Note: Latency and throughput are median [IQR] over 15 runs: three seeds and five repetitions per seed.

TABLE III: Validation-tuned model-level compression versus PTT under upper-bounded Distilled and Lite compute budgets.

		Distilled Budget			Lite Budget		
Dataset	Method	F1	GFLOPs	Lat.	F1	GFLOPs	Lat.
CSTNET TLS1.3	N-CE	0.9160	0.177	76.34	0.8261	0.075	65.38
	N-KD	0.8867	0.177	75.86	0.8484	0.075	65.40
	L2-Ch	0.9040	0.177	75.88	0.8921	0.075	64.85
	PTT	**0.9380**	0.187	**41.79**	**0.9325**	0.080	**36.12**
Cipher Spectrum	N-CE	0.8914	0.173	88.25	0.7712	0.073	70.32
	N-KD	0.8589	0.173	87.69	0.7749	0.073	70.34
	L2-Ch	0.8850	0.173	87.72	0.8393	0.073	69.74
	PTT	**0.9238**	0.182	**48.31**	**0.9136**	0.078	**38.85**

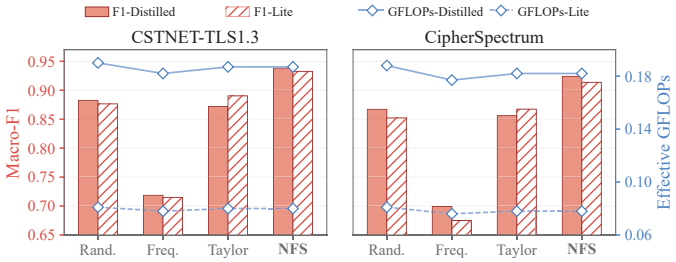


Fig. 5: Field-selector comparison under the same TopK+k closure and $K = 5$.

Top-5 sets achieve average pairwise Jaccard similarities of 0.4286 ± 0.1431 on CSTNET-TLS1.3 and 0.3982 ± 0.1536 on CipherSpectrum. The similar overlaps on both datasets indicate that the exact Top-K selections remain seed-sensitive rather than invariant. Nevertheless, repeatedly selected protocol fields form a more stable semantic core, while fields near the selection boundary may vary. Specifically, on CSTNET-TLS1.3, several fields, including IP length, TCP length, TLS cipher suite, and TLS extension type, are consistently retained across all seeds. The learned salience therefore provides a useful compression signal with a stable semantic core rather than a deterministic ranking of every protocol field.

c) Protocol Closure and Case Study: After fixing the selected fields, we examine whether their original hierarchical relations should also be retained. Selected Fields Only (SFO) keeps only the selected real fields. Selected + Ancestors (S+A) additionally retains the same ancestor nodes used by TopK+k, but removes their original parent–child relations. TopK+k retains both the required ancestors and their original hierarchical edges. Path completeness equals one only when every node and hierarchical edge on the original Full-PTG path of a selected field is retained.

TABLE IV: PTG-D retention schemes using the same NFS-selected fields and $K = 5$. Here, 'A.N.' denotes ancestor nodes, 'H.E.' denotes hierarchical edges, and 'Comp.' denotes path completeness, respectively.

Dataset	Scheme	A.N.	H.E.	Comp.	F1-D	F1-L
CSTNET TLS1.3	SFO	✗	✗	0.00	0.9010	0.8894
	S+A	✓	✗	0.00	0.8834	0.8636
	TopK+k	✓	✓	**1.00**	**0.9380**	**0.9325**
Cipher Spectrum	SFO	✗	✗	0.00	0.8860	0.8663
	S+A	✓	✗	0.00	0.8679	0.8380
	TopK+k	✓	✓	**1.00**	**0.9238**	**0.9136**

Table IV shows that retaining additional ancestor nodes alone is insufficient. S+A keeps the same ancestor nodes as TopK+k but performs below even SFO when their original hierarchical relations are removed. In contrast, TopK+k achieves complete protocol paths and the highest Macro-F1 on both datasets. Together with Fig. 5, this result separates the two roles of PTG-native compression: learned field salience determines *what* protocol evidence is retained, while closure determines *how* that evidence remains connected to its native hierarchy.

Fig. 6 gives a concrete example using the TLS-Record-Header expert. Five real fields are selected by the learned salience, while closure restores three intermediate protocol nodes required by their original paths. For example, `tls.recordheader.extension.type` is retained together with `tls`, `tls.recordheader`, and `tls.recordheader.extension`, rather than becoming an isolated field. The resulting PTG-D reduces this expert from 19 to 8 PTG schema nodes, excluding the fixed sink, while preserving the protocol context of all selected fields.

d) Structural Reduction and Field Budget: We finally measure the actual graph reduction produced by the default $K = 5$ setting. PTG-D removes more than half of the graph

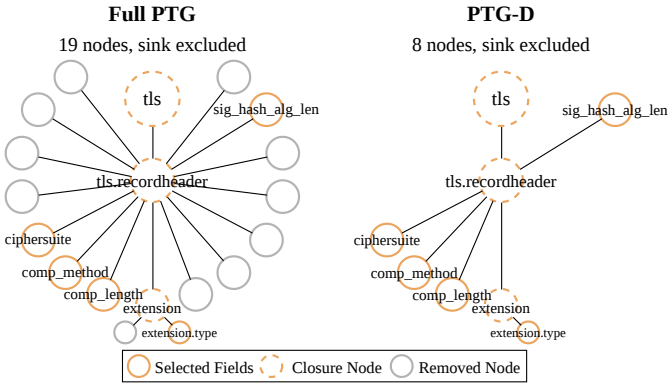


Fig. 6: PTG-D construction for the TLS-Record-Header expert.

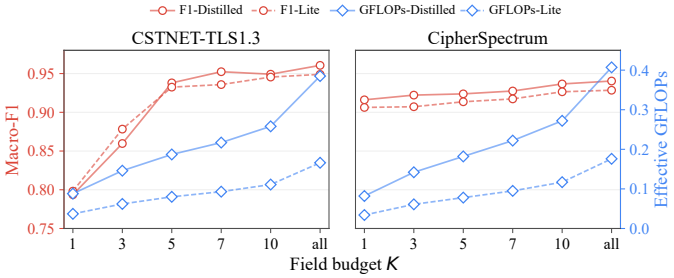


Fig. 7: Effect of the field budget K in TopK+k closure.

structure on both datasets. CSTNET-TLS1.3 is reduced from 82 to 38 nodes and from 72 to 28 hierarchical edges, while CipherSpectrum is reduced from 88 to 38 nodes and from 78 to 28 edges. These correspond to node reductions of 53.66%–56.82% and edge reductions of 61.11%–64.10%, while path completeness remains 1.00.

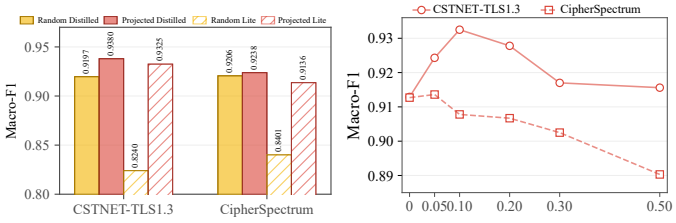
Compression is not uniform across experts: node retention ranges from 25.93% to 88.89% on CSTNET-TLS1.3 and from 25.93% to 85.71% on CipherSpectrum. PTT therefore fixes a semantic field budget rather than a graph-retention ratio. The final PTG-D size is determined by field availability and the protocol paths required by the selected fields.

Fig. 7 further varies this field budget. To emphasize the actual performance–computation trade-off, we report Macro-F1 and GFLOPs and omit parameter counts, which change only marginally with K.

On CSTNET-TLS1.3, the performance gain begins to saturate around $K = 5$: larger budgets increase graph computation substantially while providing smaller additional gains. CipherSpectrum benefits more consistently from additional fields and therefore admits higher-accuracy operating points at $K = 10$ or $K = \text{all}$. We use $K = 5$ as the common compact setting because it substantially reduces graph computation on both datasets while retaining strong classification performance.

F. Effect of Cross-Stage Transfer

After obtaining PTG-D, we further investigate how knowledge is transferred from the structurally compressed teacher



(a) Effect of channel-projection initialization. (b) Effect of the logits-KD weight (λ_{KD}) for PTT-Lite.

Fig. 8: Effect of Cross-Stage Transfer.

TABLE V: Effect of the PTT-Lite hidden dimension d_L.

Dataset	d_L	F1	Params (M)	GFLOPs
CSTNET-TLS1.3	16	0.9275	0.178	0.054
	20	0.9325	0.186	0.080
	28	0.9318	0.206	0.146
CipherSpectrum	16	0.8883	0.473	0.053
	20	0.9136	0.481	0.078
	28	0.9346	0.501	0.143

to the deployment-oriented Lite model. In the following experiments, the target PTG topology and model width are kept unchanged.

a) Effect of Channel-Projection Initialization: Fig. 8(a) compares random initialization with structure-aligned projection. Projection improves Full-to-Distilled Macro-F1 by 0.0183 and 0.0032, and yields larger gains of 0.1085 and 0.0735 for Distilled-to-Lite. The larger Lite gains indicate that transferred channels mainly stabilize adaptation to an already fixed compact architecture.

b) Effect of Flow-Level Logits Distillation: Fig. 8(b) evaluates whether flow-level teacher predictions provide additional guidance after width reduction. On CSTNET-TLS1.3, $\lambda_{\text{KD}} = 0.10$ raises Macro-F1 from 0.9129 to 0.9325. CipherSpectrum benefits only marginally at 0.05 and degrades under larger teacher weights. Logits distillation is therefore complementary and dataset-dependent rather than the main source of structural-compression gains.

c) Effect of Width Selection: Table V varies d_L within the fixed PTG-D topology. On CSTNET-TLS1.3, increasing d_L from 16 to 20 improves Macro-F1, while $d_L = 28$ adds computation without further gain. CipherSpectrum needs more capacity: $d_L = 16$ degrades clearly, and $d_L = 28$ reaches 0.9346 at 0.143 GFLOPs. We use $d_L = 20$ as a common balanced point, while $d_L = 28$ remains an accuracy-oriented option for CipherSpectrum.

VI. Conclusion

We present Pruned Traffic Trees (PTT), a protocol-structured ETC model family that makes native protocol structures directly compressible. PTT determines what protocol evidence to retain through learned field salience and how to preserve its structural context through TopK+k closure, constructing compact PTG-Ds without breaking native protocol paths. Together with sparse execution and cross-stage

adaptation, PTT provides practical Full, Distilled, and Lite operating points. Experiments under flow-disjoint and SII-masked settings demonstrate strong Macro-F1 with substantially reduced graph computation and CPU latency. Future work will investigate end-to-end parsing overhead, optimize graph execution, and extend PTT to broader protocols.

REFERENCES

[1] E. Papadogiannaki and S. Ioannidis, "A survey on encrypted network traffic analysis applications, techniques, and countermeasures," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021.

[2] Z. Guan, G. Xiong, G. Gou, Z. Li, M. Cui, and C. Liu, "Bapm: block attention profiling model for multi-tab website fingerprinting attacks on tor," in *Proceedings of the 37th Annual Computer Security Applications Conference*, 2021, pp. 248–259.

[3] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-cnn: A data-efficient website fingerprinting attack based on deep learning," *arXiv preprint arXiv:1802.10215*, 2018.

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

[5] M. S. Rahman, P. Sirinam, N. Mathews, K. G. Gangadhara, and M. Wright, "Tik-tok: The utility of packet timing in website fingerprinting attacks," *arXiv preprint arXiv:1902.06421*, 2019.

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

[7] Z. Jin, T. Lu, S. Luo, and J. Shang, "Transformer-based model for multi-tab website fingerprinting attack," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 1050–1064.

[8] 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," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 5420–5427.

[9] M. Shen, K. Ji, Z. Gao, Q. Li, L. Zhu, and K. Xu, "Subverting website fingerprinting defenses with robust traffic representation," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 607–624.

[10] Y. Zhu, J. Tao, H. Wang, L. Yu, Y. Luo, T. Qi, Z. Wang, and Y. Xu, "Dgnn: Accurate darknet application classification adopting attention graph neural network," *IEEE Transactions on Network and Service Management*, 2023.

[11] Y. Tian, S. Pei, X. Zhang, C. Zhang, and N. V. Chawla, "Knowledge distillation on graphs: A survey," *ACM Computing Surveys*, vol. 57, no. 8, pp. 1–16, 2025.

[12] Y. Luo, J. Tao, L. Yu, and G. Cheng, "Treat traffic like trees: A semantic-preserving hierarchical graph-based expert framework for encrypted traffic analysis," *arXiv preprint arXiv:2606.04517*, 2026.

[13] N. Wickramasinghe, A. Shaghaghi, G. Tsudik, and S. Jha, "Sok: Decoding the enigma of encrypted network traffic classifiers," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 1825–1843.

[14] Y. Zhao, G. Dettori, M. Boffa, L. Vassio, and M. Mellia, "The sweet danger of sugar: Debunking representation learning for encrypted traffic classification," in *Proceedings of the ACM SIGCOMM 2025 Conference*, 2025, pp. 296–310.

[15] X. Xiao, S. Wang, G. Hu, Q. Li, K. Mao, X. Luo, B. Zhang, and S. Xia, "Rbljan: Robust byte-label joint attention network for network traffic classification," *IEEE Transactions on Dependable and Secure Computing*, 2024.

[16] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," *arXiv preprint arXiv:1708.06376*, 2017.

[17] 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," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 633–642.

[18] X. Qiu, G. Cheng, W. Zhu, D. Niu, and N. Fu, "Dual-channel interactive graph transformer for traffic classification with message-aware flow representation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 1, 2025, pp. 685–693.

[19] K. Feng, C. Li, Y. Yuan, and G. Wang, "Freekd: Free-direction knowledge distillation for graph neural networks," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 357–366.

[20] Z. Guo, C. Zhang, Y. Fan, Y. Tian, C. Zhang, and N. V. Chawla, "Boosting graph neural networks via adaptive knowledge distillation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 6, 2023, pp. 7793–7801.

[21] C. Huo, D. Jin, Y. Li, D. He, Y.-B. Yang, and L. Wu, "T2-gnn: Graph neural networks for graphs with incomplete features and structure via teacher-student distillation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 4339–4346.

[22] L. Yu, S. Pei, L. Ding, J. Zhou, L. Li, C. Zhang, and X. Zhang, "Sail: Self-augmented graph contrastive learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8927–8935.

[23] H. He, J. Wang, Z. Zhang, and F. Wu, "Compressing deep graph neural networks via adversarial knowledge distillation," in *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 2022, pp. 534–544.

[24] W. Zhang, X. Miao, Y. Shao, J. Jiang, L. Chen, O. Ruas, and B. Cui, "Reliable data distillation on graph convolutional network," in *Proceedings of the 2020 ACM SIGMOD international conference on management of data*, 2020, pp. 1399–1414.

[25] J. Guo, D. Chen, and C. Wang, "Alignahead: Online cross-layer knowledge extraction on graph neural networks," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.

[26] C. Wang, S. Zhou, K. Yu, D. Chen, B. Li, Y. Feng, and C. Chen, "Collaborative knowledge distillation for heterogeneous information network embedding," in *Proceedings of the ACM web conference 2022*, 2022, pp. 1631–1639.

[27] W. Zheng, E. W. Huang, N. Rao, S. Katariya, Z. Wang, and K. Subbian, "Cold brew: Distilling graph node representations with incomplete or missing neighborhoods," in *International Conference on Learning Representations*, 2022.

[28] Z. Zhang, Z. Luan, Q. Li, Z. Qi, K. Li, Y. Jiang, and Z. Yuan, "Sentinelx: A lightweight malicious traffic detection system based on programmable switches," in *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 2025, pp. 1–10.

[29] C. Fu, Q. Li, E. Bertino, and K. Xu, "Training with only 1.0‰ samples: Malicious traffic detection via cross-modality feature fusion," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, 2025, pp. 3930–3944.

[30] J. Lu, K. Yu, Y. Huang, Z. Wei, J. Yin, and X. Liu, "Seadgat: A heterogeneous graph attention network with cross distillation for encrypted traffic classification," *IEEE Internet of Things Journal*, 2026.

[31] X. Lian, Y. Zheng, Y. Liu, F. Zhou, C. Peng, and X. Gao, "Contextual masking distillation for network traffic anomaly detection," *IEEE Transactions on Information Forensics and Security*, 2026.

[32] C. Riquelme, J. Puigcerver, B. Mustafa, M. Neumann, R. Jenatton, A. Susano Pinto, D. Keysers, and N. Houlsby, "Scaling vision with sparse mixture of experts," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8583–8595, 2021.

[33] W. Cai, J. Jiang, F. Wang, J. Tang, S. Kim, and J. Huang, "A survey on mixture of experts in large language models," *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[34] J. Zhou, C. Sun, M. Shen, S. Yu, and Q. Xuan, "Traffic-moe: A sparse foundation model for network traffic analysis," *arXiv preprint arXiv:2601.00357*, 2026.

[35] Q. He, X. Fu, and L. Zhang, "Trafficmoe: Heterogeneity-aware mixture of experts for encrypted traffic classification," *arXiv preprint arXiv:2603.29520*, 2026.

[36] J. Qin, X. Han, Y. Li, D. Han, W. Qiao, X. Yang, Z. Cui, B. Jiang, and Z. Lu, "Trafficmoe: Adaptive multi-perspective feature fusion for enhancing malicious traffic general detection capability," in *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2026, pp. 14 242–14 246.

[37] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *International conference on learning representations*, 2017.