Computer Modeling in Engineering & Sciences

Tech Science Press

https://doi.org/10.32604/cmes.2026.083972

ARTICLE

# LLM-Driven Cross-Flow Modeling for Network Attack Traffic Detection

**Aoran Huang**[1,2,*], **Sinuo Zhang**[1,2], **Haoxiang Zhu**[1,2], **Xiaojing Fan**[1,2] **and Huachun Zhou**[1,2,*]

[1]School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China
[2]National Engineering Research Center for Advanced Network Technologies, Beijing Jiaotong University, Beijing, China
*Corresponding Authors: Aoran Huang. Email: huangaoran@bjtu.edu.cn; Huachun Zhou. Email: hchzhou@bjtu.edu.cn

Received: 14 April 2026; Accepted: 05 June 2026; Published: 27 July 2026

**ABSTRACT:** In Future Mobile Internet and convergence application scenarios, existing network attack traffic detection methods are insufficient in characterizing cross-flow correlations and structural dependencies during the attack process, and therefore still have limited generalization ability in complex scenarios and unknown attack identification tasks. To address this issue, this paper proposes a cross-flow modeling large language model framework, which extends the traditional detection paradigm based on single-flow features to joint modeling oriented toward cross-flow context and relational structure. Specifically, this paper constructs cross-flow context through flow sorting, grouping, and cross-group sampling, and combines an inter-flow relation matrix with a dual-branch embedding mechanism to achieve structured representation and relation-aware modeling of network traffic; at the model level, by removing the causal mask and introducing a relation-aware bias into bidirectional self-attention, the representation capability of the large language model for complex attack behaviors and potential inter-flow dependencies is enhanced. Experimental results show that the proposed method overall outperforms traditional machine learning and deep learning baseline models, and demonstrates better stability and accuracy in tasks such as fine-grained classification, unknown attack identification, and cross-scenario generalization. Ablation experiments further verify the effectiveness of the proposed cross-flow context construction, dual-branch embedding, and relation-aware LLM adaptation, demonstrating that each component contributes to the overall detection performance and generalization ability. Our work shows that, after targeted structural adaptation, large language models can effectively serve non-text security tasks such as network traffic analysis, thereby supporting AI-driven security modeling for Future Mobile Internet environments.

**KEYWORDS:** Large language models; network traffic detection; network security; attention mechanism

## 1 Introduction

With the rapid development of 5G/6G, cloud-edge collaboration, and Internet of Things services, both the scale of network traffic and the complexity of interactions have continued to increase [1]. Attack behaviors have accordingly evolved to exhibit characteristics such as high concurrency, multi-stage progression, cross-protocol activity, and cross-service coupling. Although traditional detection methods based on statistical features of individual flows can identify some known attacks, they often struggle to characterize the coordinated behaviors among different flows involved in the same attack campaign [2]. Meanwhile, in real-world network environments, substantial differences in collection conditions, background services, attack toolchains, and noise proportions often lead to distribution shifts between training data and deployment scenarios, which limits the generalization capability of existing models across scenarios, datasets, and attack types [3]. Therefore, improving both cross-flow relationship modeling and model generalization in complex network environments has become a key challenge in malicious traffic detection.

In recent years, large language models have demonstrated strong capabilities in long-range dependency modeling and contextual representation learning, offering a new perspective for complex network traffic analysis [4]. Transformers model global interactions through the self-attention mechanism [5], while pretrained models such as BERT further demonstrate the transfer potential of bidirectional contextual representation learning [6]. Beyond text, studies such as Time-LLM [7], LLMTime [8], and One Fits All [9] have shown that, with appropriate adaptation, large models can be transferred to non-language tasks such as time series analysis. In the network traffic domain, studies including ET-BERT [10], NetGPT [11], and Lens [12] also indicate that pretrained Transformers and large-model methods provide a new technical path for traffic representation learning. However, directly feeding network traffic into LLMs still faces three core challenges. First, network traffic is a non-text modality, and thus requires a flow token representation that preserves statistical information while remaining suitable for LLM encoding. Second, key attack indicators often lie in cross-flow interactions rather than being fully observable within an isolated flow; without effective relational constraints, the global attention mechanism of LLMs can easily be distracted by irrelevant flows. In this case, flows sharing endpoints, ports, protocols, or similar statistical patterns should be distinguished from flows that merely co-occur in the same input context. Third, full-parameter fine-tuning for specific security tasks is costly and impractical for real-world deployment [13].

To address these issues, this paper proposes an LLM-driven cross-flow modeling framework for malicious traffic detection. The method first constructs cross-flow context from the perspective of cross-flow semantic organization instead of treating context as a simple temporal concatenation of adjacent flows. Specifically, raw flows are preprocessed, ordered, and grouped, and a cross-group sampling strategy is adopted to form contextual sequences that preserve both similarity and diversity, enabling the model to observe horizontal relationships among different flows within the same input. On this basis, inter-flow relations are further defined in terms of endpoints, ports, protocols, and statistical-feature similarity, and the multidimensional association between any two flows is compressed into an indexable relation matrix for subsequent use by the attention mechanism. To resolve the mismatch between network traffic and the input space of language models, this paper designs a flow tokenization and dual-branch embedding mechanism. One branch uses an MLP to extract stable statistical combination features within a single flow, while the other branch uses a Transformer to model cross-flow dependencies in context. The two types of information are then fused into flow token representations suitable for LLM processing. Building on this design, the paper further proposes a relation-aware LLM adaptation method, which removes the unidirectional causal mask used in conventional language models and introduces a learnable bias term indexed by the relation matrix into bidirectional self-attention. As a result, attention allocation depends not only on token content similarity but also on structural priors across flows. In addition, the paper adopts a lightweight training strategy that freezes the LLM backbone and trains only the newly added embedding layers, relation-bias parameters, and classification head, thereby balancing modeling capability and deployment cost.

To verify the effectiveness of the proposed method, experiments are conducted in two representative scenarios: DDoS attack detection and malicious encrypted traffic identification. In the DDoS scenario, a self-constructed dataset is built in a laboratory environment, covering five major attack categories and multiple specific attack types. The public CICDDoS2019 dataset is further introduced as a supplement, and during training the traffic ratio between the two data sources is controlled at approximately 1:1 to mitigate the bias caused by differences in collection environments. During evaluation, two cross-dataset validation settings, namely "self-constructed → public" and "public → self-constructed," are further designed to better reflect the cross-domain generalization requirements of real-world deployment. In addition, binary detection and fine-grained type analysis are also conducted in the malicious encrypted traffic scenario. Experimental results show that the proposed framework achieves more stable overall performance than comparison

models such as Random Forest, CNN, and LSTM, particularly in fine-grained type identification, zero-shot attack detection, and cross-dataset generalization. Ablation studies further demonstrate that the proposed cross-flow context construction, dual-branch embedding, and relation-aware LLM adaptation all contribute substantially to the overall detection performance and generalization ability.

The main contributions of this paper are as follows:

- A cross-flow context construction method for malicious traffic detection is proposed. By means of ordering, grouping, and cross-group sampling, it forms semantically targeted contextual sequences and further defines cross-flow relations based on endpoints, ports, protocols, and statistical similarity, allowing structural information among flows to be directly utilized by the subsequent attention mechanism.
- A relation-aware LLM adaptation mechanism is proposed. By introducing relation bias terms into bidirectional self-attention, attention allocation is jointly constrained by token content features and structural priors, thereby improving the model's ability to characterize multi-source, multi-stage, and coordinated attack behaviors while avoiding costly full-parameter fine-tuning.
- An evaluation scheme closer to real deployment scenarios is established. Based on the self-constructed DDoS dataset covering five major attack categories, the public CICDDoS2019 dataset is incorporated for mixed training and cross-dataset evaluation, and the validation scope is further extended to malicious encrypted traffic detection and fine-grained type analysis, so as to assess the model's accuracy and generalization from multiple perspectives.

Extended statement: This paper is an extended version of the authors' preliminary work, "An LLM-Based Method for the Analysis of Multiple Network Behaviors," which has been accepted for presentation at MobiSec 2025 (Sapporo, Japan, December 16–18, 2025). Compared with the earlier version, the present manuscript further refines the design of cross-flow context organization, dual-branch embedding, and relation-aware attention, and also includes additional cross-dataset generalization experiments, more complete ablation studies, and expanded experimental analysis.

## 2 Related Work

### *2.1 Network Attack Traffic Detection Based on Single-Flow Features*

Early studies on network attack detection largely took the individual flow as the basic unit of analysis, treating each flow as an independent sample and performing classification or anomaly detection based on features such as duration, packet-length distribution, directional statistics, ports, and protocols. This line of research established the basic paradigm for traffic detection: even without complete payload parsing, a considerable portion of malicious behavior can still be identified using only flow-level statistical features. Representative studies include Kitsune [14] for online anomaly detection and Deep Packet [15] for encrypted traffic classification. Subsequent work further introduced CNNs, residual networks, and temporal fusion structures to improve the representation of local patterns and dynamic changes, such as TNN-IDS [16] for MQTT scenarios and Res-TranBiLSTM [17], which combines residual learning with bidirectional temporal modeling. These studies show that feature learning centered on individual flows remains one of the most fundamental and effective technical routes in network traffic detection.

The above studies mainly address one question: when the model observes only a single flow, how much attack-related discriminative information can be extracted from it. Whether in traditional machine learning or subsequent deep learning methods, the core objective has been to improve the ability to recognize intra-flow patterns. For tasks with relatively fixed attack types and clear semantic boundaries, this modeling assumption is often valid. For this reason, single-flow detection methods remain important baselines in intrusion detection, encrypted traffic classification, and lightweight deployment scenarios.

However, single-flow modeling generally assumes that the label of a flow is determined primarily by its own features. When attack behavior manifests as multi-source coordination, cross-protocol linkage, or multi-stage penetration, the truly critical information often does not reside entirely within any individual flow, but is instead distributed across the relationships among multiple flows. In other words, existing single-flow methods are not incapable of learning useful flow representations; rather, their modeling granularity usually stops at the features of the flow itself, making it difficult to further explore inter-flow relationships. The present work builds on the advantages of flow-level modeling while extending the object of analysis from isolated flows to cross-flow contexts with relational structure.

### *2.2 Sequence Modeling and Transformer-Based Traffic Analysis*

To overcome the limitations of independent single-flow modeling, recent studies have begun to organize network traffic into sequences and to use temporal models or Transformers to capture contextual dependencies among flows. The introduction of the Transformer made self-attention a core tool for modeling long-range dependencies [5]. Subsequent studies, such as TabTransformer [18], FT-Transformer [19], and SAINT [20], demonstrated on tabular and structured data that attention mechanisms can effectively generate contextualized feature representations. These ideas were then introduced into traffic analysis, where network traffic was no longer treated merely as a static feature vector, but rather as a token sequence of packets, datagrams, or flow sequences that could be encoded and modeled in context [21].

On this basis, a number of representative studies based on Transformer architectures or pretraining frameworks have emerged in networking and security. ET-BERT [10] demonstrated through large-scale pretraining on unlabeled encrypted traffic that a bidirectional Transformer can learn datagram representations with transferability. More recently, EAPT further introduced adversarial pre-trained Transformers for encrypted traffic classification, showing that pretraining-based traffic representation remains an active direction for encrypted traffic analysis [22]. NetGPT [11] attempted to understand and model network traffic from a generative perspective. ITCT [23] introduced the pretraining paradigm into IoT traffic classification. In addition, other studies have further expanded the application scope of Transformers in traffic analysis through multi-task learning [24], hybrid-granularity feature fusion [25], graph-structured modeling [26], and visualized traffic representations [27]. Recent surveys on GNN-based intrusion detection also indicate that graph learning is effective for capturing structural associations among network entities and flows, but its integration with large-model attention remains underexplored [28]. Overall, this line of research has shifted the focus from single-flow feature extraction to flow-context representation learning.

However, existing sequence modeling methods still exhibit two notable limitations. First, many methods construct context primarily according to temporal order, fixed windows, or local adjacency, yet such sequences do not necessarily align with attack semantics and may easily place a large number of accidentally co-occurring but weakly related flows into the same context. Second, although self-attention can automatically learn dependency relationships, attention weights are still largely driven by token content, with insufficient use of structural cues such as shared endpoints, port correspondences, protocol consistency, and statistical similarity. By contrast, this paper does not simply organize flows into sequences and feed them into a Transformer. Instead, it first constructs a cross-flow context that is more consistent with attack semantics, and then encodes inter-flow relationships as learnable biases that directly participate in attention computation. Put differently, this paper is concerned not only with which flows are similar in content, but also with which flows are more likely to jointly constitute an attack pattern in terms of structure and behavior.

### *2.3 LLM Adaptation and Foundation Models for Non-Text Data*

Another research line closely related to this paper concerns the transfer and adaptation of large models to non-text data. BERT and LLaMA [29] respectively represent the development of bidirectional pretrained Transformers and open foundation language models. Subsequent studies have further shown that such models are not limited to natural language processing. Works such as Time-LLM, LLMTime, and One Fits All indicate that, under frozen or lightweightly adjusted pretrained backbones, language models can be reprogrammed or transferred to non-language scenarios such as time series. The core issue addressed by this line of research is no longer confined to how to use the conversational ability of LLMs, but rather how to continue exploiting the contextual relational capability embedded in pretrained models for the analysis of relationships among flows when both the input modality and the task objective have changed.

In the network traffic domain, similar ideas have also begun to emerge. In addition to ET-BERT, NetGPT, and Lens, graph-based traffic foundation model research [30] has attempted to establish a unified pretraining framework over flow-level spatiotemporal relationships. At the same time, systematic surveys of Transformers and LLMs for intrusion detection have pointed out that this direction is developing rapidly, yet still exhibits clear gaps in structural awareness [13], adaptation mechanisms [31], and security-semantic modeling [32]. This suggests that the key to truly applying large models to cybersecurity tasks does not lie in whether the conversational capability of LLMs is used, but in whether the input organization and relationship modeling mechanisms are redesigned for the special modality of traffic.

In summary, the differences between this paper and existing studies are mainly reflected at three levels in Table 1. Traditional single-flow methods mainly focus on statistical or local representations within individual flows. Sequence modeling and Transformer-based methods introduce contextual modeling, but their contexts are often constructed through temporal windows or local adjacency, which may not fully reflect attack-related structural relations. Pretrained and large-model-based traffic methods further improve representation transfer, but they still provide limited explicit modeling of relations such as shared endpoints, ports, protocols, and statistical similarity. Compared with these studies, the proposed method jointly models cross-flow context, flow representation, and inter-flow structural relations within a unified LLM-based framework. Rather than directly applying a pretrained model to isolated flows, it introduces context construction, dual-branch embedding, and relation-aware attention to align large-model modeling with the structural characteristics of network traffic.

**Table 1:** Comparison of DDoS detection methods from the perspective of flow modeling.

| Method | Representative Studies | Problem Addressed | Limitation |
|---|---|---|---|
| Single-flow Modeling Methods | [14–17] | Able to identify attack patterns from the statistical characteristics or local representations of individual flows. | Usually assume that flows are mutually independent, making it difficult to capture collaborative attack behaviors. |
| Sequence Modeling and Transformer-based Methods | [5,18–20,23–27] | Introduce contextual modeling and long-range dependency representation learning. | The construction of context is relatively simple, and the structural relationships among flows are not sufficiently exploited. |

(Continued)

**Table 1 (continued)**

| Method | Representative Studies | Problem Addressed | Limitation |
|---|---|---|---|
| Pre-trained Model-based Methods | [6–12,21,22,29,30] | Demonstrate that pre-trained models can be transferred to non-text traffic scenarios. | Lack dedicated adaptation mechanisms for modeling cross-flow structural relationships. |
| Proposed Method | — | Jointly models flow content, flow context, and inter-flow relationships. | — |

## 3 Scheme

### *3.1 Framework Overview*

Fig. 1 illustrates the overall system architecture. Taking network flows as input, the framework processes the data through three stages, namely cross-flow context construction, flow token embedding, and LLM-based flow-context analysis, with the final output produced by a classification head. After flow extraction and preprocessing, the captured traffic is represented as a set of flows. Each flow is organized as a structured input, as shown in Fig. 1, and the set of flows is denoted by $\mathcal{F} = \{f_1, f_2, \cdots, f_N\}$. For each flow $f_i$, its corresponding traffic feature representation is denoted by $x_i \in \mathbb{R}^B$, where $B$ represents the dimensionality of the traffic features. For each input sample, it is further represented as a contextual sequence $\mathcal{C}_t = \{c_1, c_2, \cdots, c_g\}$ consisting of $g$ flows, where each $c_j \in \mathcal{C}_t$ denotes one flow. In addition, to characterize the relationships among different flows in the sequence, a relation matrix $\mathcal{R}_t = \{0, \cdots, M-1\}^{g \times g}$ of size $g \times g$ is constructed, where $R_{ij} \in \mathcal{R}_t$ denotes the index of the relation type between flow $c_i$ and flow $c_j$. $M$ denotes the total number of all possible relation types. Different relation types are determined by a set of predefined rules, such as whether two flows share the same source or destination address, port, or protocol, or whether they exhibit high similarity in statistical features. Based on these rules, a relation category can be assigned to any pair of flows and further mapped to a unique index, thereby yielding a discrete representation of inter-flow relationships.

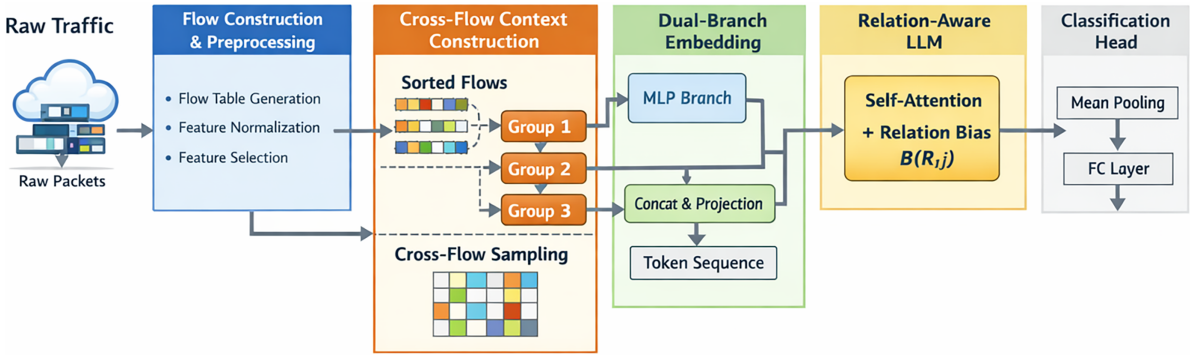



**Figure 1:** Overall architecture of the proposed LLM-driven cross-flow modeling framework.

Given the cross-flow contextual sequence $\mathcal{C}_t$ centered on the target flow and its corresponding relation matrix $\mathcal{R}_t$, the proposed model predicts the class label of the target flow by learning a contextual representation from related flows. The overall model can be formulated as

$$g_\Theta : (\mathcal{C}_t, \mathcal{R}_t) \rightarrow \hat{y}_t \tag{1}$$

where $\hat{y}_t$ denotes the predicted class label of the target flow, and $\Theta$ denotes the set of trainable model parameters.

The overall framework is described in the following three parts:

Cross-Flow Context Construction: The raw flow set is organized into contextual sequences with semantic similarity, and an implicit relation index $R$ is defined for any pair of flows within the sequence.

Flow Tokenization and Dual-Branch Embedding: The features of each flow are mapped into a flow token $e_j \in \mathbb{R}^D$, forming a flow-token sequence $\mathrm{E}_t \in \mathcal{R}^{g \times D}$.

Relation-Aware LLM Adaptation: Based on bidirectional self-attention combined with learnable attention biases determined by $\mathcal{R}_t$, contextual representations are produced and then passed through a two-layer fully connected classification head to obtain $\hat{y}_t$.

### *3.2 Cross-Flow Context Construction*

The objective of this subsection is to construct the traffic set $\mathcal{F}$into a contextual sequence $\mathcal{C}_t$ and to generate a relation index matrix $\mathcal{R}_t$ that can be directly used by the attention mechanism. The sequence generation process consists of four steps, namely traffic preprocessing, similar-flow aggregation, group-wise extraction, and relation-matrix determination, and on this basis the relation indices between flows are defined. Fig. 2 presents the detailed process of cross-flow context construction. First, the raw flows are sorted according to statistical features and divided into multiple groups. Based on the grouping results, a cross-group sampling strategy is adopted to construct contextual sequences, in which flows with similar ranks in different groups are aligned, thereby forming a semantically meaningful cross-flow representation.

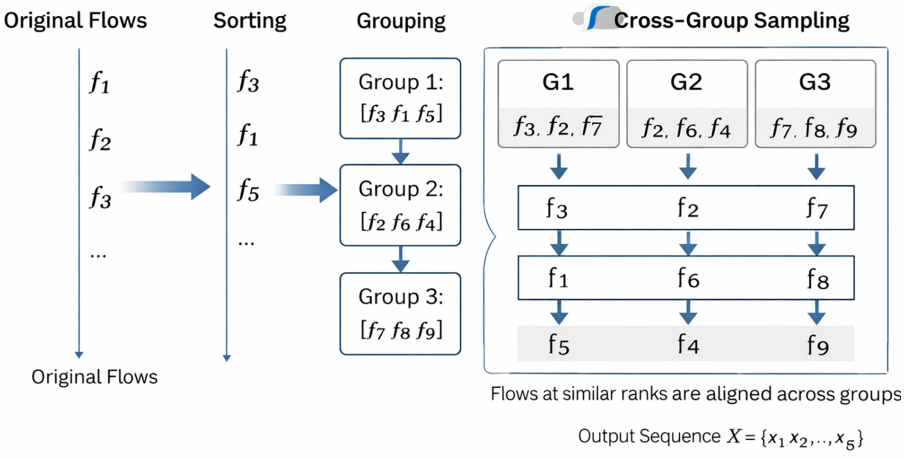



**Figure 2:** Cross-flow context construction via sorting, grouping, and cross-group sampling.

#### *3.2.1 Traffic Preprocessing*

After the traffic data are converted into flow-table features, the raw features need to be standardized to reduce the impact of redundant features on model training. This process includes two steps: normalization and feature selection. For continuous features, Min-Max normalization is used to scale them into the interval $[0,1]$. For the $j$-th feature, the normalization process can be expressed as:

$$x_j^{norm} = \frac{x_{i,j} - \min(x_{\cdot,j})}{\max(x_{\cdot,j}) - \min(x_{\cdot,j})} \tag{2}$$

Subsequently, to reduce the interference of redundant features, a random forest model is first used to evaluate the importance of each feature, and the importance of the $j$-th feature is denoted by $I_j$. The features are ranked according to their importance scores, and the subset of features with higher scores is retained as $F_{imp} = \{j | I_j \geq \tau_1\}$, where $\tau_1$ is the importance threshold. On the filtered feature set $F_{imp}$, the Pearson correlation coefficient between any two features is further calculated as

$$\rho_{p,q} = \frac{\mathrm{cov}(x_{\cdot,p}, x_{\cdot,q})}{\sigma_p \sigma_q} \tag{3}$$

where $\sigma_p$, $\sigma_q$ denote the standard deviations of features $p$ and $q$, respectively. When $\rho_{p,q} \geq \tau_2$, the two features are considered highly correlated, and only one of them is retained, yielding the final feature set $\mathcal{F}_{final}$.

After the above processing, the final feature input matrix $X = [x_1, x_2, \cdots, x_N] \in \mathbb{R}^{N \times B}$ is obtained, where $B = \left|\mathcal{F}_{final}\right|$ denotes the dimensionality of the selected features.

### *3.2.2 Similar-Flow Aggregation*

To form semantically similar context, the flow sequence is first sorted so that flows with higher similarity become adjacent in the sequence. The sorting function for each flow is defined as $\kappa(f_i)$, and the sorting is performed according to the following attributes:

$$\kappa(f_i) = (SrcPort_i,\ DstPort_i,\ Proto_i, L_i^{fwd}, L_i^{bwd}, \bar{L}_i^{fwd}, \bar{L}_i^{bwd}) \tag{4}$$

where $L_i^{fwd}$, $L_i^{bwd}$ denote the total forward and backward packet lengths, and $\bar{L}_i^{fwd}$, $\bar{L}_i^{bwd}$ denote the average forward and backward packet lengths. Based on $\kappa$, all flows are sorted to obtain the reordered input feature matrix, denoted by $\tilde{X} = [\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_N] \in \mathbb{R}^{N \times B}$. Through this sorting strategy, flows with similar ports, protocols, and directional statistical features are placed closer together in the sequence, thereby providing a semantically consistent basis for subsequent cross-group extraction.

This sorting strategy is used only as a prior for organizing candidate contexts, rather than for fully recovering attack chains or multi-stage attack semantics. It reduces the chance of grouping many weakly related flows into the same context, while the subsequent relation matrix and relation-aware attention further determine the actual dependency strength among flows.

### *3.2.3 Group-Wise Extraction*

To construct a comparable cross-flow contextual sequence, the sorted flow sequence is further grouped and reorganized. Let the grouping parameter be $G$; then the length of each group is defined as

$$S = \left\lceil \frac{N}{G} \right\rceil, N' = G \cdot S \tag{5}$$

When $N' > N$, padding is required. This is achieved by replicating the features of the last flow in the sorted sequence, which can be expressed as $\tilde{x}_{N+1} = \tilde{x}_{N+2} = \cdots = \tilde{x}_{N'} = \tilde{x}_N$. In this way, the padded feature matrix $\tilde{X}' \in \mathbb{R}^{N' \times B}$ is obtained. It is then sequentially divided into $G$ sub-sequence groups, denoted by $X^{grp} \in \mathbb{R}^{G \times S \times B}$, $X_{g,s}^{grp} = \tilde{x}_{(g-1)S+s}$. Since the flows within each group are similar in the sorting sense, each group can be understood as a local set of homogeneous flows.

If only the flows within a single group are used to construct a sequence, the contextual information will be overly local, making it difficult to characterize the differences among different attack behaviors. At the same time, after flow sorting, tokens are constructed. Considering that the working mechanism of

LLMs focuses on the correlations among preceding and following tokens, a cross-group extraction strategy is further designed to maximize the mining of inter-flow relationships. Specifically, flows from different groups are combined to construct more relational contextual sequences. For each position index $s \in \{1, 2, \cdots, S\}$, the $s$-th flow is taken from each group and concatenated in group order, yielding

$$X^{(s)} = [X_{1,s}^{grp}; X_{2,s}^{grp}; \cdots; X_{G,s}^{grp}] \in \mathbb{R}^{G \times B} \tag{6}$$

The corresponding flow sequence can be denoted by $\mathcal{C}^{(s)} = [c_1^{(s)}, c_2^{(s)}, \cdots, c_G^{(s)}]$. During both training and inference, each $X^{(s)}$ is treated as an independent input sample. For notational convenience, any constructed sequence is hereafter denoted as

$$\mathcal{C}_t = [c_1, c_2, \cdots, c_g], g = G \tag{7}$$

Fig. 2 illustrates this cross-group sampling process, in which flows at the same position in different groups are merged to form a new sequence element. Through this group-wise extraction strategy, horizontal comparison among flows can be achieved based on the detailed ordering information within each group after sorting. This ensures that the flows in each sequence come from different groups, so that each sequence contains both similarity and difference information, allowing the model to observe the differences among different flows within the same context. Moreover, this design helps ensure that different relation types among flows can simultaneously appear during the subsequent construction of the relation matrix $\mathcal{R}_t$, thereby providing the LLM with richer relational information for learning.

### *3.2.4 Relation-Matrix Determination*

After obtaining the contextual sequence $\mathcal{C}_t$, if the model relies only on the features of the flows themselves, it is difficult to distinguish truly related flows from flows that merely appear together by chance. For example, multiple flows within the same attack usually share similar communication endpoints or protocol characteristics, whereas flows from different attacks often lack such consistency. Therefore, it is necessary to represent the relational information among flows in the sequence through relation coefficients, so as to further characterize the associations among different flows in the sequence. The process of relation encoding and matrix construction is shown in Fig. 3. For each pair of flows, multiple relations are first evaluated, including endpoint similarity, port consistency, protocol consistency, and feature similarity. These relations are then aggregated and encoded as indices, so that pairwise relations can be efficiently represented in matrix form.

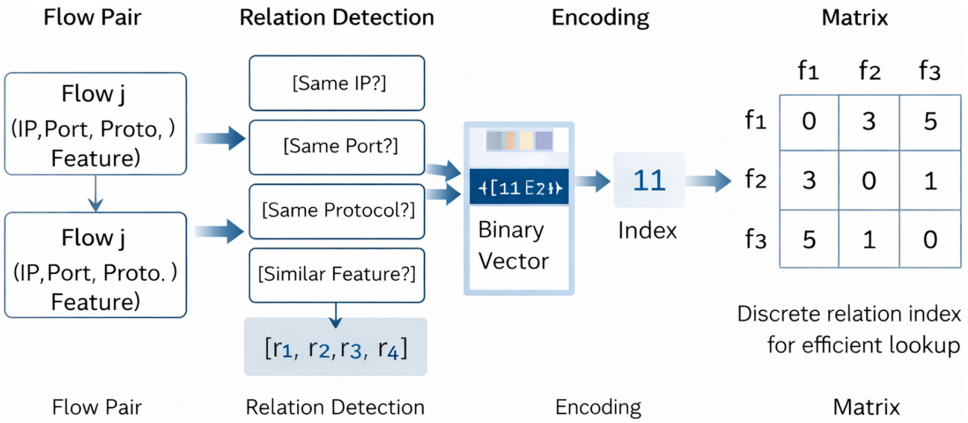



**Figure 3:** Relation encoding and matrix construction for pairwise flow interactions.

For any two flows $c_i$ and $c_j$, interpretable rules are used to determine whether a certain relation exists between them. Specifically, four types of relations are considered: endpoint correlation, where two flows share the same source IP or destination IP and may involve the same attack source, victim, or communication entity; port correlation, where two flows point to the same service port and may correspond to similar service targets or attack entry points; protocol consistency, where two flows use the same protocol and may reflect similar communication behavior; and feature similarity, where two flows are similar in statistical features such as packet length, duration, or directional patterns. These four relations can be expressed as follows:

$$\begin{aligned}
&\phi_{ep}(c_i, c_j) = \{(SrcIP_i = SrcIP_j) \vee (DstIP_i = DstIP_j)\} \\
&\phi_{port}(c_i, c_j) = \{DstPort_i = DstPort_j\} \\
&\phi_{proto}(c_i, c_j) = \{Proto_i = Proto_j\} \\
&\phi_{sim}(c_i, c_j) = \{s(x_{c_i}, x_{c_j}) \geq \tau\}
\end{aligned} \tag{8}$$

where $s(\cdot, \cdot)$ denotes the similarity function and $\tau$ is the threshold. Each of the above relations can be regarded as a binary judgment result. All relations are then combined into a relation vector, which reflects the association between two flows across different dimensions and can be expressed as

$$r_{ij} = [\phi_{ep}(c_i, c_j), \phi_{port}(c_i, c_j), \phi_{proto}(c_i, c_j), \phi_{sim}(c_i, c_j)] \in \{0, 1\}^4 \tag{9}$$

To facilitate efficient querying of related relationships by the subsequent attention mechanism, the above binary relation vector is further mapped into an integer index

$$R_{ij} = \sum_{k=1}^{4} r_{ij}^{(k)} \cdot 2^{k-1}, M = 16 \tag{10}$$

where $R_{ij}$ denotes the relation type index between flows $c_i$ and $c_j$, and 16 is the total number of all possible relation combinations. In this way, multiple relation judgments are compressed into a unique indexable identifier for direct subsequent use.

By calculating the corresponding relation index $R_{ij}$ for all flow pairs $(c_i, c_j)$ in the sequence, the relation matrix can be obtained as

$$\mathcal{R}_t = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1g} \\ R_{21} & R_{22} & \cdots & R_{2g} \\ \vdots & \vdots & \ddots & \vdots \\ R_{g1} & R_{g2} & \cdots & R_{gg} \end{bmatrix} \in \{0, 1, \cdots, 15\}^{g \times g} \tag{11}$$

This matrix fully describes the relational structure between any two flows in the sequence. Finally, the output of the cross-flow modeling module is represented as

$$(\mathcal{C}_t, \mathcal{R}_t) \tag{12}$$

### *3.3 Flow Tokenization and Dual-Branch Embedding*

After obtaining the contextual sequence $\mathcal{C}_t$ and its corresponding features, the features need to be transformed into input representations suitable for LLM processing. To this end, each flow is mapped into a flow-token vector, and a sequence representation is further constructed, whose corresponding feature matrix can be denoted by $X_t = [x_{c_1}, x_{c_2}, \cdots, x_{c_g}] \in \mathbb{R}^{g \times B}$. The objective is to learn a mapping that converts raw flow features into token representations $E_t \in \mathbb{R}^{g \times D}$ that can be understood by the LLM. To fully exploit different

types of information contained in flow features, a dual-branch embedding structure is adopted, consisting of an MLP branch for feature combination and a Transformer branch for contextual inter-flow association analysis. If only the MLP is used, the relationships among flows may be overlooked; if only the Transformer is used, the stability of single-flow features may be weakened. Therefore, the MLP branch is used to focus on the internal structure of individual flows, while the Transformer branch is used to capture cross-flow dependencies. In this way, the dual-branch structure can account for both statistical features and sequential relationships, achieving a complementary effect.

**MLP branch:** This branch is used to model the feature-combination relationships within an individual flow. Through a two-layer fully connected network, the raw features are mapped into a low-dimensional space, which can be expressed as

$$H^{(m)} = \sigma(X_t W_{m1} + b_{m1}) W_{m2} + b_{m2} \in \mathbb{R}^{g \times d} \tag{13}$$

where $\sigma(\cdot)$ denotes the nonlinear activation function. The role of this branch is to perform nonlinear combinations of the statistical features of each flow and extract stable local representations.

**Transformer branch:** This branch is used to model the dependencies among different flows within the sequence. The features are first mapped into the encoder input space

$$U = X_t W_{e0} + b_{e0} \in \mathbb{R}^{g \times d} \tag{14}$$

and then fed into the Transformer, whose output is denoted by $V = f_{TE}(U) \in \mathbb{R}^{g \times d}$. The output is finally mapped to the same dimensionality as that of the MLP branch:

$$H^{(t)} = V W_{e1} + b_{e1} \in \mathbb{R}^{g \times d} \tag{15}$$

This branch is able to capture the interaction relationships among different flows in the sequence and provide contextual information for subsequent cross-flow modeling.

To integrate the information from the two branches, their outputs are concatenated along the feature dimension and then mapped into the input space of the LLM, which can be expressed as

$$E_t = [H^{(m)} || H^{(t)}] W_p + b_p \in \mathbb{R}^{g \times D} \tag{16}$$

where $W_p \in \mathbb{R}^{(2d) \times D}$. The final $E_t$ thus obtained is the token-sequence representation fed into the LLM.

### *3.4 Relation-Aware LLM Adaptation*

For the constructed contextual sequence $\mathcal{C}_t$ and the corresponding relation matrix $\mathcal{R}_t$, the attention mechanism of the LLM needs to be adapted to both, so that it can exploit not only the feature similarity of the flow sequence, but also the relational characteristics among flows. The overall structure of the proposed relation-aware attention mechanism is shown in Fig. 4. Unlike conventional self-attention, which assigns weights solely according to content similarity, a relation bias determined by $\mathcal{R}_t$ is introduced, enabling the model to further identify which flows are more likely to be related during attention computation. As a result, flows with stronger semantic or structural connections are assigned higher attention weights.

It should be emphasized that this paper adopts bidirectional attention, that is, no causal mask is used, so that information exchange is allowed between any two flows in the sequence. On this basis, the relation bias is further used to adjust the interaction strength between different pairs of flows.

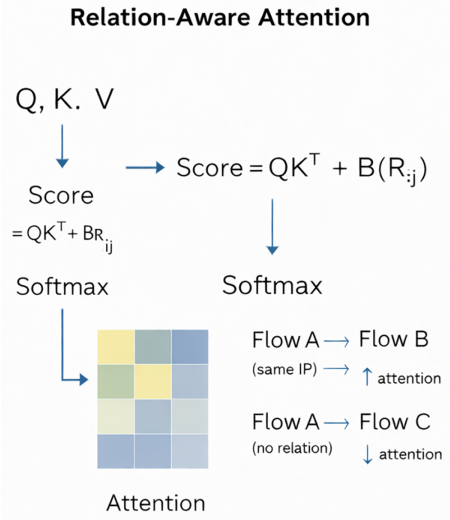



**Figure 4:** Relation-aware attention mechanism with structural bias for cross-flow modeling.

Consider a single attention head in one layer of the LLM. Given the input token representation $E_t \in \mathbb{R}^{g\times D}$, linear transformations are first applied to obtain the query, key, and value:

$$Q = E_t W_Q, K = E_t W_K, V = E_t W_V \tag{17}$$

Under the bidirectional setting, no causal mask is used, and only the Padding Mask is retained.

For any positions $i$ and $j$, the corresponding bias value $B_{ij} = b_{R_{ij}}$ is obtained by looking up the index $R_{ij}$ in the relation matrix $\mathcal{R}_t$ from a learnable parameter vector. Each relation type corresponds to one weight, which is used to adjust the attention strength between two flows. Before attention is computed, the attention score is modified according to whether the two flows are related. For example, the attention scores of flow pairs sharing endpoints or protocols are enhanced, while unrelated flow pairs retain relatively low weights. Based on standard scaled dot-product attention, the relation bias is added to the similarity computation:

$$A = \text{softmax}\left(\frac{\text{QK}^T}{\sqrt{d_k}} + B + M\right) \tag{18}$$

The final output is denoted by $Attn(E) = AV$, where $\text{QK}^T/\sqrt{d_k}$ represents feature-based similarity, $B$ represents the correction based on structural relations, and $M$ is used to mask padding. All three have dimensions of $g \times g$. After stacking multiple layers of such attention and feed-forward networks, the input sequence is updated layer by layer, and the final output of the LLM is denoted by $Z \in \mathbb{R}^{g\times D}$, where each row $Z_i$ corresponds to the representation of the $i$-th flow in the sequence after contextual information has been integrated. At this point, $Z$ is no longer an independent feature representation of a single flow, but already contains contextual information resulting from interactions with other flows.

To obtain the prediction for the target flow, average pooling is applied to the hidden states of all flows in the contextual sequence, so as to aggregate the sequence-level contextual information into a fixed-length vector:

$$z = Pool(Z) = \frac{1}{g}\sum_{i=1}^{g} Z_i \tag{19}$$

Based on the aggregated contextual representation, the final class label of the target flow is predicted through a lightweight fully connected classification head. Specifically, a two-layer structure is used:

$$h = \sigma(W_1 z + b_1)$$
$$\hat{y} = \mathrm{softmax}(W_2 h + b_2) \tag{20}$$

where $W_1 \in \mathbb{R}^{d_h \times D}$ is used to map the representation into an intermediate space, $W_2 \in \mathbb{R}^{C \times d_h}$ is used to output the class distribution, and $C$ denotes the number of classes.

### *3.5 Training Strategy and Complexity*

In the preceding sections, the complete modeling process from raw traffic features to final prediction results has been established, including context construction, relation modeling, feature embedding, and the relation-aware attention mechanism. Building on this foundation, this section further describes the training strategy of the model and its overall computational cost.

Since the parameters of the LLM are frozen, a single-stage joint training strategy is adopted, in which all newly introduced modules are optimized simultaneously in one training process, without any additional pretraining or multi-stage training. Specifically, the trainable parameters of the model include $\Theta = \Theta_{emb} \cup \Theta_{bias} \cup \Theta_{head}$. Among them, $\Theta_{emb}$ corresponds to the dual-branch embedding and projection parameters, which are used to map flow features into token representations; $\Theta_{bias}$ corresponds to the relation-bias parameter $b$, which is used to regulate attention allocation; and $\Theta_{head}$ corresponds to the parameters of the classification head. Under the default setting, the backbone parameters of the LLM remain frozen and are used only as a feature transformer.

Under the supervised learning setting, standard cross-entropy loss is adopted for training. For the $n$-th sample, let the prediction result be $\hat{y}_n$ and the ground-truth label be $y_n$; then the overall loss can be expressed as

$$\mathcal{L}(\Theta) = -\sum_{n=1}^{N}\sum_{c=1}^{C} y_{n,c} \log \hat{y}_{n,c} \tag{21}$$

From the overall pipeline, the computational complexity of the proposed framework is mainly concentrated in two stages, namely context modeling and attention computation. Given a contextual sequence of length $g$, the main computational cost of self-attention is $\mathcal{O}(g^2 D)$, which arises from the computation of the attention-weight matrix and the weighted summation. On this basis, introducing the relation bias does not change the overall complexity. This is because the relation-bias matrix $B$ is obtained by querying the relation-index matrix $\mathcal{R}_t$, and the complexity of its construction is $\mathcal{O}(g^2)$. Similarly, the computational complexity of calculating the relation matrix $\mathcal{R}_t$ is also based on pairwise combinations of flows within the sequence: $\mathcal{O}(g^2)$. It should be noted that $g$ here does not denote the original number of flows $N$, but rather the contextual length obtained after group-wise extraction. Since this process divides the global traffic into multiple local sequences and models each sequence independently, both relation computation and attention computation are constrained within a relatively small scale. Although the complexity within a single sequence is quadratic, the group-wise extraction strategy distributes the overall computational burden across multiple small-scale sequences, thereby ensuring the feasibility of the method in practical applications.

It should also be noted that the above analysis mainly describes the additional computational cost introduced by relation modeling and context construction. Since the framework still uses a frozen 8B-parameter LLM backbone, its inference cost is higher than that of lightweight baselines.

## 4 Evaluation

This section evaluates the proposed framework from two perspectives: DDoS attack traffic analysis and malicious encrypted traffic identification. The experiments are designed to verify the effectiveness of the three core components introduced in Section 3, namely cross-flow context construction, dual-branch embedding, and relation-aware LLM adaptation, under standard, cross-domain, and zero-shot-like settings. Together, these results examine whether modeling beyond single-flow statistical features can provide more robust detection and generalization in complex network environments.

### *4.1 Datasets and Data Construction*

#### *4.1.1 Self-Constructed Dataset*

To comprehensively evaluate the performance of the model in complex network environments, this paper combines a self-constructed dataset with public datasets. Using only laboratory data makes it easier to control the attack generation process and label quality, but it can also lead to overly homogeneous distributions. Relying solely on public datasets, by contrast, may fail to fully reflect the multi-scenario, multi-stage, and fine-grained attack behaviors emphasized in this study. Therefore, this paper adopts a "self-constructed–public–fusion" data construction strategy. This preserves the controllability and interpretability of the experimental environment while introducing richer distributional variation through public data, thereby making the experimental conclusions more reliable.

Specifically, this paper builds a multi-scenario DDoS attack dataset on a virtualization platform to simulate attack behaviors in real network environments. Multiple Ubuntu 18.04 virtual machines are deployed on the VMware vSphere platform, each configured with 8 CPU cores, 16 GB of memory, and a 32 GB hard disk, and a three-layer network topology of "attack domain–ingress router–target domain" is established. This setup not only supports the construction of reproducible attack scenarios, but also enables unified observation of communication relationships among different hosts at the network ingress point, so that the resulting traffic data better reflect real deployment settings in which attacks are observed from the network side rather than from a single-host perspective.

Within this environment, five typical categories of DDoS attacks are simulated. First, in the Low-rate DDoS scenario, 4 of the 8 hosts act as attack machines and use SlowHTTPTest to generate slow headers, slow body, slow read, and shrew attacks, while 2 web servers serve as victim targets to create persistent low-rate attack traffic. Second, in the Application-layer DDoS scenario, the attack machines use tools such as Hulk, Webbench, and GoldenEye to simulate HTTP-Flood, CC, and HTTP GET/POST flooding. Third, in the Distributed Reflection DDoS scenario, Scapy is used to construct spoofed requests that trigger reflection servers to generate reflected attack traffic such as Memcached, TFTP, Chargen, NTP, SNMP, and SSDP. Fourth, in the Botnet-based DDoS scenario, bot hosts communicate with command-and-control servers, and the attack carriers include Ares, BYOB, Zeus, and Mirai. Fifth, in the Network-layer DDoS scenario, the attack machines use hping to launch SYN Flood, ACK Flood, and UDP Flood attacks. These five attack categories cover different protocol layers, control modes, and attack rhythms, and thus provide a good reflection of the heterogeneity of complex malicious traffic.

To avoid a self-constructed dataset containing only malicious communications and therefore becoming detached from real network environments, benign traffic is also generated. On the one hand, Python socket is used to simulate normal requests in scenarios such as 5G public services, smart homes, and machine-type communications. On the other hand, normal traffic samples from CICIDS2017 are introduced as a supplement. These CICIDS2017 samples are used only to enrich the benign traffic pool, rather than as an independent evaluation dataset. All traffic is captured at the ingress router node using tcpdump to

generate PCAP files, and CICFlowMeter is then used to convert the raw packets into flow-table features. This processing ensures that all subsequent experiments can be conducted uniformly at the flow level and remains consistent with the procedure described in Section 3.2 for traffic preprocessing, feature sorting, group-wise sampling, and relation-matrix construction.

The self-constructed dataset was generated in a controlled laboratory environment and does not involve real personal user communications. All experiments are conducted on flow-level statistical features without using plaintext payloads. Fields such as endpoints and ports are used only for relation construction and statistical modeling, rather than for identifying individuals. The public datasets are obtained from publicly available sources and used according to their public usage conditions.

#### *4.1.2 Public Dataset Fusion*

In addition to the self-constructed data, this paper also introduces the public datasets CICDDoS2019 and CICAndMal2017 as supplements. These two datasets contain various typical DDoS attack types and encrypted traffic and exhibit high diversity and complexity, making them well suited for testing the adaptability of the model under different scenario distributions. For the main DDoS detection experiments, the training data are constructed by mixing the self-constructed dataset and CICDDoS2019 at an approximately 1:1 ratio, so as to improve data diversity and enhance model robustness under standard in-domain evaluation settings. In addition, to further assess cross-domain generalization, two cross-dataset transfer settings are designed, namely self-constructed dataset → CICDDoS2019 and CICDDoS2019 → self-constructed dataset. In these transfer experiments, the model is trained on only one source dataset and directly tested on the other target dataset without mixed-domain training.

Through this fusion strategy, the self-constructed data provide controllable attack structures and clear labels, whereas the public data provide distributional diversity. Compared with relying on a single source, this setting is more helpful for improving the model's generalization ability in complex network environments. It should be noted that this paper does not equate data fusion directly with cross-domain robustness; rather, this issue is further examined through cross-source testing and zero-shot analysis in the subsequent tasks.

To clarify the data scale and class distribution, all in-domain experiments are split at the raw-flow level with an 8:2 stratified train-test ratio before cross-flow context construction. Sorting, grouping, cross-group sampling, and relation-matrix construction are then performed independently within each split. For zero-shot-like evaluation, the selected unseen attack types are removed from the training set and used only in the testing stage. The final numbers of samples used in each experimental task are reported in Table 2.

**Table 2:** Final training and testing samples used in each experimental task.

| Experimental Task | Class/Category | Total Samples | Training Samples | Testing Samples |
|---|---|---|---|---|
| Standard DDoS detection | Benign/DDoS | $2 \times 12,000$ | $2 \times 9600$ | $2 \times 2400$ |
| DDoS major-category analysis | Benign + 5 major DDoS categories | $6 \times 6000$ | $6 \times 4800$ | $6 \times 1200$ |
| Fine-grained DDoS analysis | Benign + 22 DDoS subtypes | $23 \times 2000$ | $23 \times 1600$ | $23 \times 400$ |

(Continued)

**Table 2 (continued)**

| Experimental Task | Class/Category | Total Samples | Training Samples | Testing Samples |
|---|---|---|---|---|
| Zero-shot-like detection | Known classes | 6 × 6000 | 6 × 4800 | 6 × 1200 |
| Zero-shot-like detection | Unseen attack types | 20,000 | 0 | 20,000 |
| Zero-shot-like detection | Benign samples for unseen testing | 20,000 | 0 | 20,000 |
| Cross-dataset evaluation I | Self-constructed → CICDDoS2019 | 12, 000 + 12, 000 | 12,000 | 12,000 |
| Cross-dataset evaluation II | CICDDoS2019 → Self-constructed | 12, 000 + 12, 000 | 12,000 | 12,000 |
| Encrypted malicious traffic detection | Benign/Malicious | 2 × 6000 | 2 × 4800 | 2 × 1200 |
| Encrypted malicious traffic analysis | Benign + 4 malicious families | 5 × 2000 | 5 × 1600 | 5 × 400 |

#### *4.1.3 Preprocessing and Feature Selection*

All data are processed using the same preprocessing pipeline as in Section 3. First, Min-Max normalization is applied to the features. Then, a random forest model is used to assess feature importance, and features with relatively low contribution scores are removed. Finally, Pearson correlation coefficients are calculated on the filtered feature set, and redundant features with correlation coefficients greater than 0.9 are removed, thereby reducing the interference of highly collinear features during model training.

After feature selection, the final DDoS dataset retains key features including Subflow Fwd Packets, Fwd Seg Size Min, FWD Init Win Bytes, Flow Duration, Idle Max, Packet Length Min, Fwd Packet Length Min, Fwd Header Length, Idle Min, Total Fwd Packet, Src Port, Dst Port, and Protocol. The malicious encrypted traffic dataset retains features such as time_start, tcp.out.flags, byte_dist_mean, time_end, packets.ipt.out, expire_type, dp, packets.ipt.in, tcp.in.flags, tcp.in.opts.mss, tcp.in.opt_len, byte_dist_std, ip.out.ttl, tcp.out.opt_len, bytes_out, tcp.first_seq, tcp.out.first_window_size, packets.b, tcp.in.opts.ws, entropy, and total_entropy.

After the above feature selection process, the final feature dimensionality is $B = 13$ for the DDoS detection task and $B = 21$ for the malicious encrypted traffic detection task. In the subsequent model input, $B$ denotes the final flow-feature dimensionality after feature selection for the corresponding task.

### *4.2 Experimental Settings*

Given that network attack detection tasks often involve large variations in attack patterns, potential class distribution differences, and complex deployment environments, this paper jointly adopts four evaluation metrics: Accuracy, Precision, Recall, and F1-score. Accuracy is used to characterize overall classification

correctness, Precision reflects the reliability of the model when predicting the target class, Recall measures the coverage of the target class, and F1-score provides a balanced evaluation of Precision and Recall.

To ensure representative comparisons, Random Forest (RF), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM) are selected as basic baseline models. RF represents traditional statistical learning methods and mainly characterizes the separability of single-flow statistical features. CNN represents deep models that are sensitive to local patterns. LSTM represents a typical sequence modeling method and can capture a certain degree of sequential dependency. To further cover recent attention-based, graph-based, and pretrained traffic analysis routes, RA-Transformer, GAT, and ET-BERT are also included in the comparison. RA-Transformer denotes a relation-aware Transformer trained from scratch. It uses the same cross-flow input and relation-bias setting as the proposed method, but replaces the frozen LLM backbone with a randomly initialized Transformer encoder. GAT is used as a graph-based baseline, where flows are treated as nodes and the relation matrix is used to construct graph edges. ET-BERT is adopted as a representative pretrained Transformer-based traffic model in the malicious encrypted traffic experiments, where its original task setting is more consistent with encrypted traffic classification. Overall, these baselines cover traditional single-flow statistical learning, local deep feature learning, sequential modeling, non-LLM Transformer modeling, graph-based relation modeling, and pretrained encrypted-traffic representation, thereby providing a more complete comparison for evaluating the proposed framework.

To avoid drawing conclusions from evaluation under only a single data split, this paper designs evaluation from three aspects.

- **In-domain Evaluation:** the model is trained and tested on splits drawn from the same task-specific dataset to assess the standard detection performance under a shared data distribution.
- **Cross-domain Evaluation:** the model is trained on the self-constructed dataset and tested on CICD-DoS2019, and vice versa.
- **Zero-shot-like Evaluation:** some attack types are removed from the training set and introduced only during testing, in order to assess the model's ability to recognize unseen attack patterns.

For representative neural-network-based experiments, the training process is repeated under 10 random seeds, and the main results are reported as mean ± standard deviation. Unless otherwise specified, all models are evaluated under the same training/testing splits and hardware environment.

To avoid temporal or contextual leakage during cross-flow context construction, data splitting is first performed at the raw-flow level. Sorting, grouping, cross-group sampling, and relation-matrix construction are then conducted independently within each partition. Therefore, flows in the testing set are not used to construct training contexts or training relation matrices. In addition, flows from the same attack session or the same collection time block are kept within the same split to prevent source-related contexts from appearing in both training and testing sets.

In the zero-shot-like setting, the unseen attack types are removed from the training set before context construction and appear only in the testing stage. In the cross-domain evaluation, the model is trained on one source dataset and directly tested on the other target dataset, without target-domain fine-tuning or mixed-context construction. These settings allow the three evaluation protocols to respectively examine closed-set detection, unseen attack recognition, and cross-source generalization.

The detailed implementation environment and hyperparameter settings are summarized in Table 3. All experiments are conducted on a workstation running Ubuntu 20.04. The hardware platform is equipped with a 40-core Intel® Xeon® Silver 4210R CPU and four NVIDIA GeForce RTX 3070 GPUs. The software environment uses Python 3.8, PyTorch 2.3.0 with CUDA 12.1, and Llama3-Instruct (8B) as the frozen backbone model. The backbone parameters are frozen, and only the newly added embedding modules,

relation-bias parameters, and classification head are trained. The contextual sequence length is set to 32, and the embedding dimension is aligned with the input dimension of the backbone. Adam is used as the optimizer, cross-entropy loss is adopted as the loss function, the learning rate is set to 0.001, and the batch size is set to 16. All baseline models and the proposed method are run under the same training/testing splits and hardware environment, thereby ensuring a fair comparison.

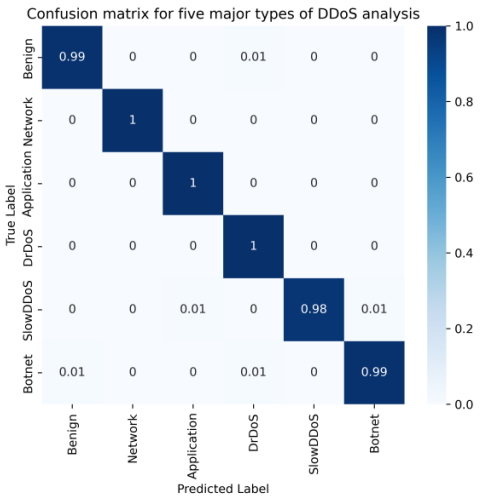
**Table 3:** Implementation details and hyperparameter settings.

| Item | Setting |
|---|---|
| Operating system | Ubuntu 20.04 |
| CPU | 40-core Intel® Xeon® Silver 4210R |
| GPU | 4 NVIDIA GeForce RTX 3070 GPUs |
| CUDA version | 12.1 |
| Programming language | Python 3.8 |
| Deep learning framework | PyTorch 2.3.0 |
| Backbone of proposed method | Llama3-Instruct (8B), frozen |
| Trainable modules | Dual-branch embedding, relation bias, classification head |
| Context length | 32 |
| Optimizer | Adam |
| Learning rate | 0.001 |
| Batch size | 16 |
| Loss function | Cross-entropy loss |
| DDoS feature dimension | $B = 13$ |
| Encrypted traffic feature dimension | $B = 21$ |
| Number of repeated runs | 10 |

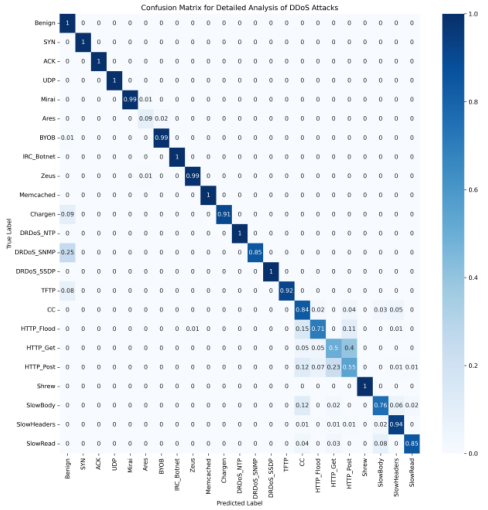
### *4.3 DDoS Detection Results*

Experiments are first conducted on the self-constructed DDoS dataset and the public CICDDoS2019 dataset under four settings, namely major-category attack analysis, fine-grained type analysis, standard DDoS detection, and zero-shot detection. The corresponding results are shown in Fig. 5 and Table 4. Since these four tasks differ in both difficulty and objective, this paper focuses not only on whether the model achieves higher numerical results, but also on what capabilities these results represent. Standard detection reflects basic recognition ability, major-category analysis reflects coarse-grained attack semantic partitioning ability, fine-grained type analysis reflects fine-grained discrimination ability, and zero-shot detection more directly reflects structural generalization ability.

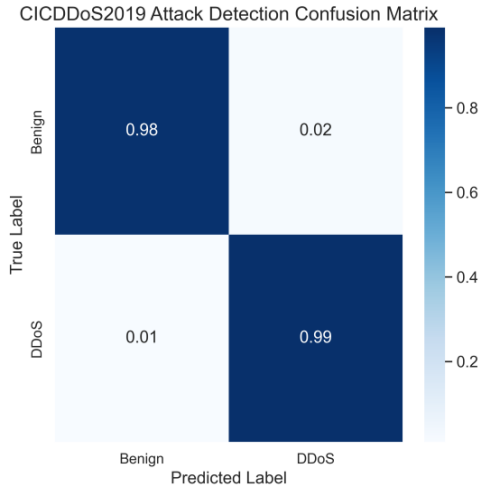
To further compare the proposed method with attention-based and graph-based models, two additional baselines are evaluated on representative DDoS tasks. RA-Transformer uses the same cross-flow input and relation-bias setting as the proposed method, but replaces the frozen LLM backbone with a randomly initialized Transformer encoder. GAT treats flows as graph nodes and constructs edges according to the relation matrix. These two baselines help distinguish the contribution of the frozen LLM backbone from that of relation-aware attention and graph-based structural modeling.



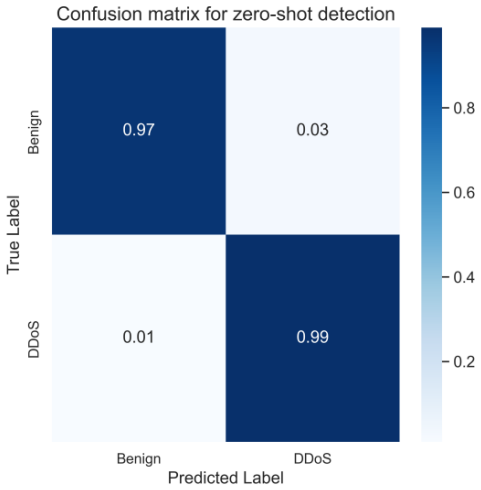

a DDoS Major-Category Attack Analysis




b DDoS Fine-Grained Attack Type Analysis




c Attack Detection on CICDDoS2019




d Zero-Shot Analysis on CICDDoS2019

**Figure 5:** Visualization results of the proposed method under different DDoS analysis settings. (**a**) DDoS major-category attack analysis; (**b**) DDoS fine-grained attack type analysis; (**c**) attack detection on the CICDDoS2019 dataset; (**d**) zero-shot analysis on the CICDDoS2019 dataset.

**Table 4:** Performance comparison on DDoS detection tasks.

| Model | Metric | DDoS Detection | DDoS Major-Category Analysis | DDoS Fine-Grained Type Analysis | Zero-Shot Detection |
|---|---|---|---|---|---|
| Our | Accuracy | 0.9912 | 0.9921 | 0.8937 | 0.9850 |
| | Precision | 0.9913 | 0.9924 | 0.9068 | 0.9854 |
| | Recall | 0.9912 | 0.9921 | 0.8937 | 0.9850 |
| | F1 Score | 0.9912 | 0.9921 | 0.8911 | 0.9850 |

(Continued)

**Table 4 (continued)**

| Model | Metric | DDoS Detection | DDoS Major-Category Analysis | DDoS Fine-Grained Type Analysis | Zero-Shot Detection |
|---|---|---|---|---|---|
| RF | Accuracy | 0.9806 | 0.9599 | 0.8022 | 0.9402 |
| | Precision | 0.9810 | 0.9487 | 0.8222 | 0.9537 |
| | Recall | 0.9805 | 0.9786 | 0.8022 | 0.9402 |
| | F1 Score | 0.9805 | 0.9601 | 0.7925 | 0.9434 |
| CNN | Accuracy | 0.9968 | 0.9945 | 0.8398 | 0.7271 |
| | Precision | 0.9968 | 0.9912 | 0.8444 | 0.7569 |
| | Recall | 0.9968 | 0.9943 | 0.8398 | 0.7271 |
| | F1 Score | 0.9968 | 0.9928 | 0.8213 | 0.7408 |
| LSTM | Accuracy | 0.9649 | 0.9936 | 0.8305 | 0.8650 |
| | Precision | 0.9671 | 0.9896 | 0.8684 | 0.8997 |
| | Recall | 0.9648 | 0.9943 | 0.8305 | 0.8650 |
| | F1 Score | 0.9649 | 0.9918 | 0.8282 | 0.8756 |

As shown in Table 5, RA-Transformer achieves competitive results under the same cross-flow input and relation-bias setting, but its performance is still lower than that of the proposed method on all three representative tasks. Specifically, compared with RA-Transformer, the proposed method improves the accuracy by 0.45, 2.06, and 0.72 percentage points on DDoS detection, fine-grained type analysis, and zero-shot detection, respectively. This suggests that the relation-aware design itself is useful, while the frozen LLM backbone brings additional gains in modeling complex contextual interactions. The GAT baseline also benefits from the explicit relation graph and achieves reasonable performance, but it is consistently lower than RA-Transformer and the proposed method, especially on fine-grained type analysis and zero-shot detection. This suggests that directly treating the relation matrix as graph edges is useful, but still less sufficient for capturing complex cross-flow dependencies than relation-aware attention within the LLM-based contextual modeling framework. Overall, these results further support the effectiveness of combining cross-flow structural priors with the contextual representation capability of the frozen LLM backbone.

**Table 5:** Additional comparison with attention-based and graph-based baselines on representative DDoS tasks.

| Method | Modeling Type | DDoS Detection Accuracy | Fine-Grained Type Analysis Accuracy | Zero-Shot Detection Accuracy |
|---|---|---|---|---|
| RA-Transformer | Non-LLM Transformer with relation bias | 0.9867 ± 0.0021 | 0.8731 ± 0.0042 | 0.9778 ± 0.0043 |
| GAT | Graph attention on relation graph | 0.9814 ± 0.0029 | 0.8587 ± 0.0069 | 0.9583 ± 0.0055 |
| Ours | Frozen LLM with relation-aware attention | 0.9912 ± 0.0013 | 0.8937 ± 0.0036 | 0.9850 ± 0.0025 |

As can be seen from Fig. 5a,c, in the major-category analysis and standard detection tasks, the decision boundaries between benign traffic and the major attack categories are relatively clear, and errors are mainly concentrated among a small number of categories with similar patterns. This indicates that in scenarios where attack semantics are relatively explicit and category boundaries are relatively clear, the proposed method can stably extract effective discriminative information from both flow-level features and contextual relationships. More importantly, Fig. 5b,d correspond to two more challenging scenarios. On the one hand, fine-grained type analysis substantially increases the number of categories and the similarity among classes. On the other hand, the zero-shot setting directly uses attack types that do not appear during training for testing. Under both conditions, the model still maintains strong recognition ability, which suggests that it learns not merely the superficial patterns of known categories in the training set, but more likely general contextual structures related to attack behavior.

As shown in Table 4, the proposed method exhibits more stable overall performance across most tasks. First, in the standard DDoS detection task, both the Accuracy and F1-score of the proposed method reach 0.9912, significantly outperforming RF and LSTM. Although CNN achieves slightly higher Accuracy in this task, this advantage does not persist in more complex scenarios. A more reasonable interpretation is that CNN has an advantage in fitting local patterns under simple binary classification settings, but this advantage depends more on a relatively stable task distribution; once the task shifts to finer-grained or out-of-distribution settings, its performance is no longer stable.

Second, in the major-category attack analysis task, the proposed method maintains all four metrics at around 0.992. Compared with RF, this indicates that relying solely on single-flow statistical features is no longer sufficient to support robust discrimination in complex scenarios. Compared with LSTM, it suggests that sequence modeling itself is not the sole source of performance improvement; more important are how the contextual sequence is constructed and whether the relationships among flows are explicitly utilized. In other words, the gain of the proposed method lies not simply in using a sequence model, but in using a sequence organization strategy that is more consistent with attack semantics.

In the fine-grained DDoS type analysis task, the proposed method achieves an Accuracy of 0.8937 and a Precision of 0.9068, both clearly higher than those of the three baseline models. This task involves many categories and fine-grained decision boundaries, and is therefore more likely to expose the limitations of a model in learning subtle differences among similar attack types. The fact that the proposed method still maintains strong overall performance under these conditions indicates that its input representation does not dilute single-flow information after contextual information is introduced. Combined with the design in Section 3.3, this suggests that the MLP branch provides the model with stable local statistical representations, while the Transformer branch supplies the contextual information needed to distinguish similar attack categories. The complementary relationship between the two is especially evident in this task.

Finally, the zero-shot detection results most clearly demonstrate the generalization value of the model. In this task, both the Accuracy and F1-score of the proposed method reach 0.9850, clearly outperforming RF, CNN, and LSTM. Since the attack types used during testing do not appear in training, this result implies that the model does not primarily rely on memorizing known attack templates, but is more likely to learn transferable structural patterns. For network attack detection, this capability is more meaningful in practice than high scores in a closed-set setting, because attacks in real environments continue to evolve and cannot be exhaustively covered by the training data.

While the above results demonstrate that our proposed method performs well in in-domain detection tasks, training and testing traffic in real-world deployments often do not originate from completely identical data distributions. Therefore, examining only in-domain settings is insufficient to fully illustrate

the model's generalization ability. Based on this, we further supplement this paper with cross-dataset transfer experiments.

### *4.4 Cross-Dataset Generalization Evaluation*

To further evaluate the generalization ability of the model, we additionally conduct cross-dataset experiments on the self-constructed DDoS dataset and the public CICDDoS2019 dataset. This setting is consistent with the cross-domain evaluation described in the Introduction and is intended to examine whether the model can still maintain stable detection performance when training and testing data come from different traffic environments.

Specifically, two transfer directions are considered: self-constructed dataset → CICDDoS2019 and CICDDoS2019 → self-constructed dataset. In the first setting, the model is trained on the self-constructed dataset and directly tested on CICDDoS2019. In the second setting, the training and testing domains are reversed. To ensure comparability, the same preprocessing pipeline and model configuration as those used in the main DDoS experiments are retained, while mixed-domain training is not used in this subsection. Instead, the model is trained on a single source dataset in each transfer setting and then directly evaluated on the other dataset without further fine-tuning.

The results are reported in Table 6. It can be seen that the proposed method achieves the highest accuracy under both transfer directions. Under the self-constructed dataset → CICDDoS2019 setting, the accuracy of the proposed method reaches 0.9418, which is higher than that of RF (0.9054), CNN (0.8421), and LSTM (0.8916). This result indicates that when the model is trained on a relatively controlled self-constructed environment and tested on a more complex public dataset, conventional methods based on single-flow statistics or local pattern learning are more easily affected by distribution shifts. In contrast, the proposed method can better preserve attack-relevant structural information across flows, thus maintaining higher detection accuracy in the transfer setting.

**Table 6:** Cross-dataset evaluation results.

| Training Dataset → Testing Dataset | Method | Accuracy |
|---|---|---|
| Self-constructed Dataset → CICDDoS2019 | RF | 0.9054 |
| Self-constructed Dataset → CICDDoS2019 | CNN | 0.8421 |
| Self-constructed Dataset → CICDDoS2019 | LSTM | 0.8916 |
| Self-constructed Dataset → CICDDoS2019 | Ours | 0.9418 |
| CICDDoS2019 → Self-constructed Dataset | RF | 0.9174 |
| CICDDoS2019 → Self-constructed Dataset | CNN | 0.8588 |
| CICDDoS2019 → Self-constructed Dataset | LSTM | 0.9043 |
| CICDDoS2019 → Self-constructed Dataset | Ours | 0.9496 |

Under the CICDDoS2019 → self-constructed dataset setting, the proposed method further achieves an accuracy of 0.9496, again outperforming RF (0.9174), CNN (0.8588), and LSTM (0.9043). Compared with the reverse transfer direction, the overall performance of all methods is slightly better in this setting. A possible reason is that CICDDoS2019 contains more diverse traffic patterns, allowing the model to learn relatively richer representations during training and thus transfer more effectively to the self-constructed dataset. Even so, the proposed method still remains the best-performing model, which suggests that its gain does not come merely from dataset differences, but from its design in cross-flow organization and structural relation modeling.

The performance drop under cross-dataset evaluation mainly reflects distribution shifts between the two data sources. The self-constructed dataset and CICDDoS2019 differ in collection environments, attack tools, background traffic, class proportions, and statistical feature distributions. These differences may change the endpoint-, port-, and statistics-based relation patterns learned by the model. Nevertheless, the proposed method still maintains the highest accuracy in both transfer directions, suggesting that explicit cross-flow relation modeling improves robustness under distribution shifts.

Overall, the cross-dataset results show that the advantage of the proposed method is not limited to in-domain testing. When the training and testing domains exhibit clear distribution differences, the model still maintains higher detection accuracy, indicating that it captures relatively transferable attack-related structural features rather than relying only on shallow dataset-specific statistical patterns. This observation is also consistent with the deployment-oriented motivation of improving cross-scenario generalization.

### *4.5 Ablation Study*

Building upon the verification of the model's overall generalization ability, a further question remains: from which key modules do the aforementioned performance improvements specifically originate? To further analyze the contribution of each major component, the ablation study is conducted on three aspects: cross-flow context construction, dual-branch embedding, and relation-aware LLM adaptation.

#### *4.5.1 Ablation on Cross-Flow Context Construction*

To verify the effectiveness of the proposed flow-sequence structure, two groups of ablation experiments are designed. In the first group, single-flow detection is adopted. Under the same flow feature vectors, no flow-sequence structure is constructed; instead, each flow is independently input into the same LLM model as used in this paper for classification. In the second group, a time-window-based flow-sequence detection method is used, in which flow sequences of length 32 are constructed according to the order of traffic arrival. The third group corresponds to the proposed method, which adopts a flow-sequence construction strategy based on similar-flow aggregation.

As shown by the results in Table 7, in the DDoS detection task, the proposed method achieves an Accuracy of 99.12%, which is 4.70 and 2.75 percentage points higher than those of the sliding-window flow-sequence method and the single-flow structure, respectively. This indicates that the performance gain does not arise merely because the model receives sequential input. The more important difference lies in how the sequence is constructed. Simply organizing traffic in temporal order introduces context, but such context does not necessarily correspond to real attack relationships. By contrast, the proposed method places greater emphasis on the aggregation of similar flows during flow-sequence construction, which is more conducive to learning structural patterns related to attack behavior.

**Table 7:** Ablation results of different flow organization strategies.

| Variant | Sequence Construction | Standard DDoS Detection Accuracy | Zero-Shot Detection Accuracy |
|---|---|---|---|
| Single-flow Input | No sequence | 0.9637 | 0.9637 |
| Time-window Sequence | Sliding window, length = 32 | 0.9442 | 0.9399 |
| Ours | Similarity-aware cross-flow sequence | 0.9912 | 0.9850 |

This point is even more evident in the zero-shot detection task. Under the single-flow detection setting, the model achieves an Accuracy of 96.37%; under the time-window flow-sequence setting, the Accuracy is 93.99%; whereas the proposed method reaches 98.50% in the zero-shot task. Since the attack types in the zero-shot setting do not appear during training, this result indicates that relying solely on single-flow statistical features, or merely constructing sequences based on temporal proximity, is insufficient to provide the model with stable generalization cues. By contrast, the flow-sequence structure proposed in this paper can more effectively preserve the contextual dependencies among flows, allowing the model to maintain strong recognition ability even when facing unseen attack types.

Overall, this group of ablation experiments shows that, for network attack traffic detection, the key issue is not simply whether a sequence is constructed, but how the sequence is constructed. The proposed cross-flow construction not only improves conventional DDoS detection performance, but also enhances the robustness of the model in zero-shot scenarios.

### *4.5.2 Ablation on Dual-Branch Embedding*

On the basis of the above validation of cross-flow context construction, we further examine the effect of the dual-branch embedding module. As described in Section 3.3, the dual-branch embedding consists of an MLP branch and a Transformer branch. The former is used to preserve and combine local statistical features of individual flows, while the latter is introduced to model contextual dependencies within the flow sequence. To verify whether the two branches are complementary, we keep the cross-flow context construction and relation-aware LLM adaptation unchanged, and only replace the embedding structure.

Three variants are considered: MLP-only, Transformer-only, and the complete Dual-branch design. Since the value of the dual-branch mechanism is more directly reflected in fine-grained discrimination among similar attack types, this ablation is conducted on the fine-grained type analysis task, which imposes higher requirements on feature representation.

The results are shown in Table 8. When only the MLP branch is used, the model achieves an accuracy of 0.8748. When only the Transformer branch is used, the accuracy is 0.8671. By contrast, the complete dual-branch design achieves the highest accuracy of 0.8937. These results indicate that neither branch alone is sufficient to fully characterize complex traffic patterns, while their combination leads to more effective representations.

**Table 8:** Ablation results of the dual-branch embedding module.

| Embedding Variant | Accuracy |
|---|---|
| MLP-only | 0.8748 |
| Transformer-only | 0.8671 |
| Dual-branch | 0.8937 |

More specifically, the relatively strong performance of MLP-only suggests that single-flow statistical features still provide an important basis for traffic discrimination. For many attack categories, basic fields and statistical patterns already contain useful information. However, when the task moves from coarse-grained detection to fine-grained attack-type classification, relying only on single-flow features makes it more difficult to distinguish categories with similar behaviors, which limits the overall accuracy.

The Transformer-only setting performs slightly worse than MLP-only, indicating that contextual modeling among flows is helpful but cannot fully replace stable local feature preservation. In other words,

the contextual dependency learned by the Transformer branch can enhance sequence representation, but without direct support from local statistical features, the model still suffers in fine-grained classification.

The complete Dual-branch design achieves the best result, showing that the two branches indeed play complementary roles. The MLP branch provides a stable local feature basis, while the Transformer branch introduces contextual dependencies across flows. Their combination allows the model to preserve the discriminative information of individual flows while further capturing subtle sequence-level differences among similar attack categories. Therefore, the gain of the dual-branch design comes from the collaboration between local statistical representation and contextual dependency modeling rather than from a simple increase in model complexity.

*4.5.3 Ablation on Relation-Aware LLM Adaptation*

After validating the context construction and embedding modules, we further analyze the effect of the relation-aware LLM adaptation module. As introduced in Section 3.4, two modifications are applied to the pretrained LLM: first, the original causal mask is removed so that flows within the same sequence can interact bidirectionally; second, a structural bias derived from the relation matrix is added to the attention scores to explicitly enhance information exchange among related flows. To separately examine the contribution of these two designs, we keep the cross-flow context construction and dual-branch embedding unchanged and compare three attention settings: Causal Mask + No Relation Bias, Bidirectional + No Relation Bias, and Bidirectional + Relation Bias, where the last one corresponds to the full method.

Since the value of relation-aware attention is mainly reflected in structural generalization under unseen conditions, this ablation is conducted on the zero-shot detection task. Compared with standard detection, the zero-shot setting more directly reveals whether the model has learned transferable structural patterns across flows rather than only memorizing known attack categories.

The results are presented in Table 9. Under the Causal Mask + No Relation Bias setting, the model achieves an accuracy of 0.9713. After removing the causal mask while still not using relation bias, the accuracy increases to 0.9782. When relation bias is further introduced, the accuracy reaches 0.9850, which is the best result among the three settings. This clearly shows that both components of the proposed adaptation are beneficial and that their effects are progressive.

**Table 9:** Ablation results of relation-aware LLM adaptation.

| Attention Setting | Accuracy |
|---|---|
| Causal Mask + No Relation Bias | 0.9713 |
| Bidirectional + No Relation Bias | 0.9782 |
| Bidirectional + Relation Bias | 0.9850 |

The improvement from 0.9713 to 0.9782 indicates that directly retaining the unidirectional autoregressive constraint of language models is not well suited to traffic analysis. Unlike text generation, traffic detection focuses on structural dependencies among flows, and useful evidence may appear at any position of the sequence. Removing the causal mask therefore allows more sufficient interaction among flows and improves detection accuracy.

The further improvement from 0.9782 to 0.9850 shows that bidirectional interaction alone is still not enough. Without explicit structural guidance, the model can perform global interaction, but it is still difficult to distinguish truly attack-relevant inter-flow dependencies from incidental co-occurrence. By introducing

relation bias, the attention mechanism is guided by structural relations such as endpoint, port, protocol and feature similarity, enabling the model to focus more effectively on meaningful flow dependencies and thus improving zero-shot detection accuracy.

To further interpret the relation-aware mechanism, we also examine the learned relation-bias coefficients after training. Each coefficient corresponds to one relation type encoded by endpoint correlation, port correlation, protocol consistency, and feature similarity. Therefore, the learned coefficients indicate how the model adjusts attention scores for different structural relations.

As shown in Table 10, the learned coefficients provide an interpretable view of how different structural relation types influence attention allocation. The no-relation type obtains a slightly negative coefficient (−0.12), indicating that flow pairs without explicit structural relations are mildly suppressed. Among individual relations, feature similarity receives the largest positive coefficient (0.24), suggesting that statistical behavioral consistency is a strong cue for cross-flow association. Endpoint and port correlations also contribute positive biases (0.11 and 0.09), while protocol consistency alone has only a weak effect (0.03), since protocol information is relatively coarse. When feature similarity is combined with endpoint or port correlations, the coefficients increase to 0.36 and 0.33, respectively, and the combination of endpoint, port, and feature similarity reaches 0.44. This indicates that the model assigns stronger attention to flow pairs that are both behaviorally similar and structurally related. The highest coefficient appears when all four relations are satisfied (0.47), further showing that the relation-aware attention mechanism uses explicit cross-flow structural priors to adjust attention allocation rather than relying only on token-content similarity.

**Table 10:** Learned relation-bias coefficients for different relation types.

| Relation Index | Endpoint | Port | Protocol | Feature Similarity | Bias Coefficient |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | −0.12 |
| 1 | 1 | 0 | 0 | 0 | 0.11 |
| 2 | 0 | 1 | 0 | 0 | 0.09 |
| 3 | 1 | 1 | 0 | 0 | 0.22 |
| 4 | 0 | 0 | 1 | 0 | 0.03 |
| 5 | 1 | 0 | 1 | 0 | 0.15 |
| 6 | 0 | 1 | 1 | 0 | 0.13 |
| 7 | 1 | 1 | 1 | 0 | 0.25 |
| 8 | 0 | 0 | 0 | 1 | 0.24 |
| 9 | 1 | 0 | 0 | 1 | 0.36 |
| 10 | 0 | 1 | 0 | 1 | 0.33 |
| 11 | 1 | 1 | 0 | 1 | 0.44 |
| 12 | 0 | 0 | 1 | 1 | 0.28 |
| 13 | 1 | 0 | 1 | 1 | 0.39 |
| 14 | 0 | 1 | 1 | 1 | 0.37 |
| 15 | 1 | 1 | 1 | 1 | 0.47 |

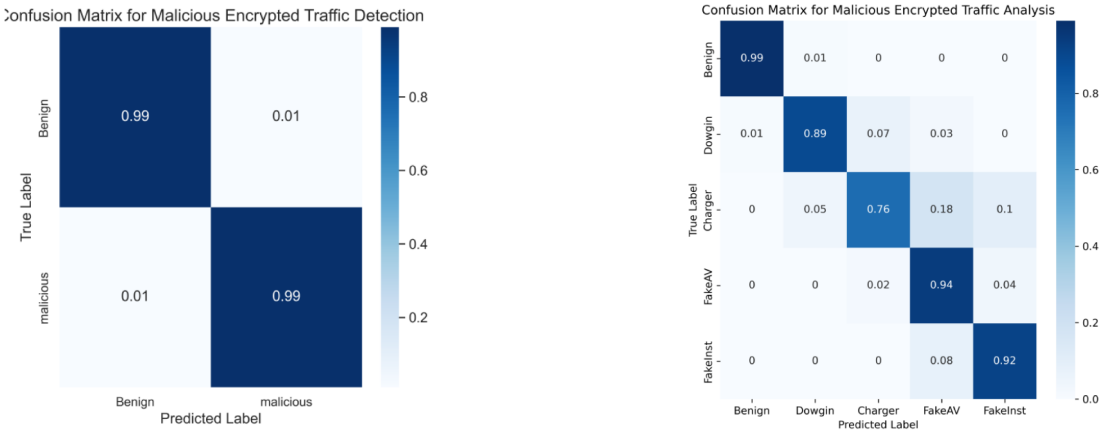
Overall, these results demonstrate that the proposed relation-aware LLM adaptation is not a simple architectural modification, but an important task-oriented adjustment for traffic analysis. By removing the autoregressive constraint that is unsuitable for this task and incorporating explicit structural bias, the model can better capture cross-flow dependencies and achieve stronger recognition ability for unseen attack patterns.

### *4.6 Encrypted Malicious Traffic Detection Results*

To further examine the adaptability of the proposed method in scenarios with limited semantic information, experiments are also conducted on a malicious encrypted traffic dataset. Compared with DDoS, malicious encrypted traffic identification depends less on explicit payload content and more on communication behavior patterns, temporal statistical features, and cross-flow relationships, making it more suitable for validating the effectiveness of dual-branch embedding and relation-aware attention.

Considering that ET-BERT is a representative pretrained Transformer model for encrypted traffic classification, it is further included as an additional baseline in the malicious encrypted traffic experiments. This comparison is conducted in this setting because the task objective and input characteristics are more consistent with the original design of ET-BERT.

As shown in Fig. 6a, in the malicious encrypted traffic detection task, the model presents a clear decision boundary between Benign and Malicious samples, with misclassifications mainly concentrated among a small number of boundary samples. This indicates that even when plaintext semantics are not directly accessible, the model can still achieve stable discrimination based on flow-level statistical features and contextual structure.




**a Malicious Encrypted Traffic Detection** **b Malicious Encrypted Traffic Analysis**

**Figure 6:** Visualization results for malicious encrypted traffic detection tasks. (**a**) Malicious encrypted traffic detection; (**b**) malicious encrypted traffic analysis.

As shown in Table 11, both ET-BERT and the proposed method achieve strong performance in the malicious encrypted traffic detection task. ET-BERT obtains slightly higher Accuracy, Recall, and F1-score, reaching $0.9989 \pm 0.0008$, $0.9991 \pm 0.0006$, and $0.9989 \pm 0.0006$, respectively, which is reasonable because ET-BERT is specifically pretrained for encrypted traffic representation. The proposed method achieves a comparable F1-score of $0.9985 \pm 0.0007$ and slightly higher Precision of $0.9987 \pm 0.0007$, indicating that cross-flow context construction and relation-aware attention can still provide competitive detection capability without relying on an encrypted-traffic-specific pretrained model. Compared with RF, CNN, and LSTM, the proposed method maintains clear advantages across all four metrics. When considered together with the DDoS results, this indicates that the value of the proposed method is not limited to a single attack scenario or dataset setting, but lies in its ability to model both flow-level statistical features and cross-flow structural relationships across different traffic detection tasks. These results suggest that the gain mainly comes from the joint use of cross-flow context, dual-branch embedding, and relation-aware attention, rather than from model scale alone. This also indirectly confirms the necessity of the designs in Sections 3.3 and 3.4.

**Table 11:** Results on encrypted malicious traffic detection.

| Task | Metric | Our | ET-BERT | RF | CNN | LSTM |
|---|---|---|---|---|---|---|
| Encrypted MaliciousTraffic Detection | Accuracy | 0.9976 ± 0.0008 | 0.9989 ± 0.0008 | 0.9904 | 0.9760 | 0.9755 |
| | Precision | 0.9987 ± 0.0007 | 0.9986 ± 0.0009 | 0.9882 | 0.9964 | 0.9517 |
| | Recall | 0.9983 ± 0.0008 | 0.9991 ± 0.0006 | 0.9815 | 0.9729 | 0.9798 |
| | F1 Score | 0.9985 ± 0.0007 | 0.9989 ± 0.0006 | 0.9848 | 0.9845 | 0.9648 |

Fig. 6b further presents the model's performance on the fine-grained malicious encrypted traffic analysis task. Overall, most samples are distributed in the diagonal region, indicating that the model can already distinguish the communication patterns of different malicious families relatively well. The small amount of confusion mainly occurs among malicious categories rather than between benign and malicious categories. This means that the model's uncertainty arises more from the fine-grained boundaries within malicious traffic than from the basic judgment of maliciousness. For detection systems, this type of error structure is usually more acceptable than misclassifying malicious traffic as benign.

It should also be clarified that these results do not imply that the model can directly generalize to all attack families without adaptation. Rather, they show that the proposed cross-flow modeling framework is applicable to different traffic detection tasks, including DDoS and malicious encrypted traffic. Broader cross-family transfer, such as direct transfer from DDoS to other intrusion or malware scenarios, still requires further evaluation.

### *4.7 Result Analysis*

By synthesizing the results from both DDoS and malicious encrypted traffic experiments, it becomes clearer why the proposed method is effective. First, the significance of cross-flow context construction lies in organizing originally independent flows into contextual units that better reflect attack semantics, so that the model no longer judges only whether a single flow is abnormal, but further examines whether multiple flows jointly constitute a certain attack pattern. This is most evident in DDoS fine-grained type analysis and zero-shot detection, because neither of these tasks can be stably solved by relying only on single-flow statistical features.

Second, the role of dual-branch embedding is to alleviate the conflict between retaining only single-flow features and relying only on contextual relationships. If the model depends entirely on the MLP, it may obtain stable local representations but fail to make full use of cross-flow dependencies. If it relies only on the Transformer, contextual modeling may dilute the key statistical information within individual flows. Through the dual-branch design, this paper balances both types of information at the input stage, which leads to stronger stability in both fine-grained DDoS analysis and malicious encrypted traffic detection.

Third, relation-aware LLM adaptation plays a key role in the stability of the model under complex and out-of-distribution tasks. In network attack detection, critical associations are not always reflected in token content similarity; more often, they appear as structural relationships such as shared endpoints, ports, protocols, and similar statistical patterns. By encoding these relationships as learnable biases and injecting them into the bidirectional attention process, the proposed method enables the model to prioritize flow pairs that are more likely to be related during attention computation. Therefore, the advantage of the proposed method in zero-shot tasks and complex multi-class tasks is not an isolated result, but is closely tied to the design in which structural relationships are explicitly incorporated into attention allocation.

Overall, the above results show that the gains of the proposed framework are consistent across different tasks and datasets, especially in scenarios that require finer discrimination or stronger generalization. This indicates that the proposed design is effective not only for improving accuracy, but also for stabilizing traffic representation under more challenging conditions.

To further examine whether the reported improvements are stable across different training runs, we report the mean and standard deviation of representative neural-network-based experiments under 10 random seeds. The selected tasks cover fine-grained classification, unseen attack detection, and malicious encrypted traffic detection, which are the main scenarios used to evaluate discrimination and generalization.

As shown in Table 12, the proposed method maintains stable performance across repeated runs on the representative tasks. The standard deviations are relatively small on DDoS fine-grained type analysis, zero-shot detection, and encrypted malicious traffic detection, indicating that the reported improvements are not caused by a single favorable random initialization. In addition, RA-Transformer shows slightly larger fluctuations on the more challenging fine-grained and zero-shot tasks, while ET-BERT remains highly stable in the encrypted traffic setting. These results further support the robustness of the proposed framework under different training runs.

**Table 12:** Stability analysis of representative experiments under multiple random seeds.

| Task | Method | Accuracy | F1 Score |
|---|---|---|---|
| DDoS Fine-Grained Type Analysis | Ours | 0.8937 ± 0.0036 | 0.8911 ± 0.0038 |
| Zero-Shot Detection | Ours | 0.9850 ± 0.0025 | 0.9850 ± 0.0026 |
| Encrypted Malicious Traffic Detection | Ours | 0.9976 ± 0.0008 | 0.9985 ± 0.0007 |
| DDoS Fine-Grained Type Analysis | RA-Transformer | 0.8731 ± 0.0042 | 0.8716 ± 0.0045 |
| Zero-Shot Detection | RA-Transformer | 0.9778 ± 0.0043 | 0.9774 ± 0.0044 |
| Encrypted Malicious Traffic Detection | ET-BERT | 0.9989 ± 0.0008 | 0.9989 ± 0.0006 |

In addition to detection performance and stability, inference time is also an important factor for practical deployment. Therefore, we further compare the runtime of different models under the same hardware environment and batch-size setting described in Table 3, as shown in Table 13.

**Table 13:** Runtime comparison under the same hardware environment.

| Method | Inference Time/1000 Samples | Setting |
|---|---|---|
| RF | 0.16 s | Standard DDoS task |
| CNN | 0.68 s | Standard DDoS task |
| LSTM | 1.12 s | Standard DDoS task |
| RA-Transformer | 3.26 s | DDoS representative tasks |
| GAT | 4.18 s | DDoS representative tasks |
| ET-BERT | 8.74 s | Encrypted traffic task |
| Ours | 52.36 s | Frozen 8B backbone |

As shown in Table 13, lightweight baselines such as RF, CNN, and LSTM require much less inference time because they operate on compact flow-level features and contain substantially fewer parameters.

RA-Transformer and GAT introduce additional costs due to attention-based context modeling and graph-structured relation propagation, while ET-BERT is slower than lightweight baselines because it uses a pretrained Transformer for encrypted traffic representation. The proposed method requires the highest inference time because it relies on a frozen 8B-parameter LLM backbone. This result is consistent with the complexity and deployment discussion in Section 3.5. Therefore, the current framework is more suitable for offline traffic analysis, high-value traffic inspection, or control-plane-assisted detection, while lightweight deployment remains an important direction for future work.

## 5 Conclusion

This paper proposes an LLM-driven cross-flow modeling framework for network attack traffic detection, aiming to improve the representation, contextual modeling, and classification of complex malicious traffic under cross-flow settings. By combining cross-flow context construction, dual-branch embedding, and relation-aware attention adaptation, the method enables the model to capture both flow-level statistical features and inter-flow structural dependencies within a unified framework.

Experiments on DDoS attack traffic and malicious encrypted traffic show that the proposed method achieves consistently strong performance, with particular advantages in fine-grained classification, unseen attack detection, and cross-dataset evaluation. These results demonstrate the value of introducing cross-flow structure into large-model-based traffic analysis.

Overall, this study shows that targeted structural adaptation is a practical way to extend large language models to non-text security tasks and provides a useful perspective for future research on LLM-based network traffic analysis.

However, the current context construction and relation definitions are still rule-based, and the frozen 8B-parameter backbone introduces relatively high inference cost. Future work will explore lightweight adaptation and broader cross-family transfer across heterogeneous intrusion and malware scenarios.

**Acknowledgement:** The authors would like to thank the National Engineering Research Center for Advanced Network Technologies, Beijing Jiaotong University, Beijing, for providing technical support, experimental facilities, and research infrastructure for this study.

**Funding Statement:** This research was funded by the National Natural Science Foundation of China, grant number U2569201, and the Fundamental Research Funds for the Central Universities, grant number 2025YJS015.

**Author Contributions:** The authors confirm contribution to the paper as follows: conceptualization, Aoran Huang, Sinuo Zhang and Huachun Zhou; methodology, Aoran Huang, Sinuo Zhang and Huachun Zhou; software, Aoran Huang, Sinuo Zhang and Huachun Zhou; validation, Aoran Huang, Sinuo Zhang and Huachun Zhou; formal analysis, Aoran Huang, Sinuo Zhang and Huachun Zhou; investigation, Aoran Huang, Sinuo Zhang and Huachun Zhou; resources, Huachun Zhou; data curation, Aoran Huang, Sinuo Zhang and Huachun Zhou; writing—original draft preparation, Aoran Huang, Sinuo Zhang and Huachun Zhou; writing—review and editing, Haoxiang Zhu, Xiaojing Fan and Huachun Zhou; visualization, Aoran Huang, Sinuo Zhang and Huachun Zhou; supervision, Huachun Zhou; project administration, Huachun Zhou; funding acquisition, Huachun Zhou. All authors reviewed and approved the final version of the manuscript.

**Availability of Data and Materials:** The data used in this study include a self-constructed dataset and publicly available datasets. The self-constructed dataset is publicly available in a GitHub repository at https://github.com/liliMpro/source_dataset. The public datasets are available from the Canadian Institute for Cybersecurity (CIC) website at https://www.unb.ca/cic/. The data supporting the findings of this study are available from these public sources.

**Ethics Approval:** This study does not involve human subject experiments or the collection of real personal user communications. The experiments use only flow-level statistical features and do not analyze plaintext payloads. Therefore, additional institutional review board approval is not applicable to this study.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References


1. Braeken A, Deac D, Nguyen TL, Gür G, Pham QV, Yapa C, et al. 6G AI security: from fundamentals to offensive and defensive landscape in 6G. IEEE Commun Surv Tutor. 2026;28:4765–99.
2. Tang D, Zuo C, Li X, Wang S, Liang W, Li K, et al. Detection and mitigation of DDoS attack against SDN flow tables. IEEE/ACM Trans Netw. 2026;34(4):4269–82. doi:10.1109/ton.2026.3674973.
3. Ouyang N, Shatte A, Lu Z, Chen C, Xiang W. Artificial intelligence in mitigating security threats for lightweight IoT devices: a survey of technologies, protocols, and future challenges. IEEE Internet Things J. 2026;13(7):14096–111.
4. Huang A, Yan J, Fan X, Zhou H. Multi-scenario cloud-edge collaborative DDoS detection in LLM-enabled AIoT. IEEE Trans Netw Sci Eng. 2026;13:3790–809. doi:10.1109/tnse.2025.3637740.
5. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Proceedings of the Advances in Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA.
6. Devlin J, Chang MW, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the NAACL-HLT; 2019 Jun 2–7; Minneapolis, MN, USA. p. 4171–86.
7. Jin M, Wang S, Ma L, Chu Z, Zhang ZY, Shi X, et al. Time-LLM: time series forecasting by reprogramming large language models [Internet]. 2023 [cited 2026 Apr 5]. Available from: https://arxiv.org/abs/2310.01728.
8. Gruver N, Finzi M, Qiu S, Wilson AG. Large language models are zero-shot time series forecasters. In: Proceedings of the Advances in Neural Information Processing Systems; 2023 Dec 10–16; New Orleans, LA, USA.
9. Zhou T, Niu P, Wang X, Sun L, Jin R. One fits all: power general time series analysis by pretrained LM. In: Proceedings of the Advances in Neural Information Processing Systems; 2023 Dec 10–16; New Orleans, LA, USA.
10. Lin X, Xiong G, Gou G, Li Z, Shi J, Yu J. ET-BERT: a contextualized datagram representation with pre-training transformers for encrypted traffic classification. arXiv:2202.06335. 2022.
11. Meng X, Lin C, Wang Y, Zhang Y. NetGPT: generative pretrained transformer for network traffic [Internet]. 2023 [cited 2026 Apr 5]. Available from: https://arxiv.org/abs/2304.09513.
12. Li X, Qian C, Wang Q, Kong J, Wang Y, Yao Z, et al. Lens: a foundation model for network traffic [Internet]. 2024 [cited 2026 Apr 5]. Available from: https://arxiv.org/abs/2402.03646.
13. Kheddar H. Transformers and large language models for efficient intrusion detection systems: A comprehensive survey. Inf Fusion. 2025;124(1):103347. doi:10.1016/j.inffus.2025.103347.
14. Mirsky Y, Doitshman T, Elovici Y, Shabtai A. Kitsune: an ensemble of autoencoders for online network intrusion detection. In: Proceedings of the Network and Distributed System Security Symposium; 2018 Feb 18–21; San Diego, CA, USA.
15. Lotfollahi M, Jafari Siavoshani M, Shirali Hossein Zade R, Saberian M. Deep packet: A novel approach for encrypted traffic classification using deep learning. Soft Comput. 2020;24:1999–2012. doi:10.1007/s00500-019-04030-2.
16. Ullah S, Ahmad J, Khan MA, Alshehri MS, Boulila W, Koubaa A, et al. Transformer neural network-based intrusion detection system for MQTT-enabled IoT networks. Comput Netw. 2023;237:110072. doi:10.2139/ssrn.4394501.
17. Wang S, Xu W, Liu Y. Res-TranBiLSTM: An intelligent approach for intrusion detection in the Internet of Things. Comput Netw. 2023;235:109982. doi:10.1016/j.comnet.2023.109982.
18. Huang X, Khetan A, Cvitkovic M, Karnin Z. TabTransformer: tabular data modeling using contextual embeddings [Internet]. 2020 [cited 2026 Apr 5]. Available from: https://arxiv.org/abs/2012.06678.
19. Gorishniy Y, Rubachev I, Khrulkov V, Babenko A. Revisiting deep learning models for tabular data. In: Proceedings of the Advances in Neural Information Processing Systems; 2021 Dec 6–14; Vancouver, BC, Canada.

20. Somepalli G, Goldblum M, Schwarzschild A, Bruss CB, Goldstein T. SAINT: improved neural networks for tabular data via row attention and contrastive pre-training [Internet]. 2021 [cited 2026 Apr 5]. Available from: https://arxiv.org/abs/2106.01342.
21. Li Q, Zhang Y, Jia Z, Hu Y, Zhang L, Zhang J, et al. DoLLM: how large language models understanding network flow data to detect carpet bombing DDoS [Internet]. 2024 [cited 2026 Apr 5]. Available from: https://arxiv.org/abs/2405.07638.
22. Zhan M, Yang J, Jia D, Fu G. EAPT: an encrypted traffic classification model via adversarial pre-trained transformers. Comput Netw. 2025;257:110973. doi:10.1016/j.comnet.2024.110973.
23. Bazaluk B, Hamdan M, Ghaleb M, Gismalla MSM, Correa da Silva FS, Batista DM. Towards a transformer-based pre-trained model for IoT traffic classification. In: Proceedings of the IEEE/IFIP Network Operations and Management Symposium; 2024 May 6–10; Seoul, Republic of Korea. p. 1–7.
24. Zhang H, Xiao X, Yu L, Li Q, Ling Z, Zhang Y. One train for two tasks: an encrypted traffic classification framework using supervised contrastive learning [Internet]. 2024 [cited 2026 Apr 5]. Available from: https://arxiv.org/abs/2402.07501.
25. Huang H, Zhou Y, Jiang F. CLA-BERT: a hybrid model for accurate encrypted traffic classification by combining packet and byte-level features. Mathematics. 2025;13(6):973. doi:10.3390/math13060973.
26. Ghadermazi J, Hore S, Shah A, Bastian ND. GTAE-IDS: graph transformer-based autoencoder framework for real-time network intrusion detection. IEEE Trans Inf Forensics Secur. 2025;20:4026–41. doi:10.1109/TIFS.2025.3557741.
27. Wasswa H, Lynar T, Nanyonga A, Abbass H. IoT botnet detection: application of vision transformer to classification of network flow traffic. In: Proceedings of the GCITC; 2023 Dec 1–3; Bengaluru, India.
28. Zhong M, Lin M, Zhang C, Xu Z. A survey on graph neural networks for intrusion detection systems: methods, trends and challenges. Comput Secur. 2024;141(3):103821. doi:10.1016/j.cose.2024.103821.
29. Touvron H, Lavril T, Izacard G, Martinet X, Lachaux M-A, Lacroix T, et al. LLaMA: open and efficient foundation language models [Internet]. 2023 [cited 2026 Apr 5]. Available from: https://arxiv.org/abs/2302.13971.
30. Van Langendonck L, Castell-Uroz I, Barlet-Ros P. Towards a graph-based foundation model for network traffic analysis [Internet]. 2024 [cited 2026 Apr 5]. Available from: https://arxiv.org/abs/2409.08111.
31. Alwhbi IA, Zou CC, Alharbi RN. Encrypted network traffic analysis and classification utilizing machine learning. Sensors. 2024;24(11):3509. doi:10.3390/s24113509.
32. Sharma A, Lashkari AH. A survey on encrypted network traffic: a comprehensive survey of identification/classification techniques, challenges, and future directions. Comput Netw. 2025;257:110984. doi:10.1016/j.comnet.2024.110984.