An LLM-Based Method for the Analysis of Multiple Network Behaviors*

Sinuo Zhang†, Aoran Huang, and Huachun Zhou

Beijing Jiaotong University, Beijing, China
sinuo0716@126.com

Abstract

In recent years, machine learning and deep learning methods have been extensively applied in network behavior analysis tasks. These methods typically create specific neural network models tailored to specific tasks and exhibit limited generalizability for unknown sample types. Drawing inspiration from the recent successful implementation of large language models (LLMs) across various disciplines, this paper investigates the application of LLMs in the domain of network behavior analysis. The objective is to utilize LLMs to capture the relationships between network flows and augment the model's detection capabilities and generalization ability. To this end, this paper proposes a model architecture capable of analyzing multiple network behavior analysis tasks and constructs a flow sequence structure to fully leverage the advantages of LLMs in modeling contextual relationships. DDoS attack detection and malicious encrypted flow identification are regarded as the typical application scenarios in this paper. To evaluate the proposed model, self-built laboratory DDoS attacks data, the public CICDDoS2019 dataset, and encrypted flow generated by malicious software are employed. The experimental results demonstrate that the proposed model exhibits significant advantages in the analysis of DDoS attacks and provides preliminary validation of the feasibility of encrypted flow detection. The model's superior detection capabilities are evident when compared to traditional neural network methods.

Keywords: network behavior analysis, LLM, flow sequences, DDoS, encrypted flow

1 Introduction

Network behavior refers to the multi-dimensional characteristics exhibited by users during activities such as data transmission and resource access. It captures the interaction logic of network flows and the behavioral intent of users. In the field of cybersecurity, the primary objective of network behavior analysis is to captures effective patterns from mixed flows, identify malicious behavior, and prevent potential threats.

Currently, the majority of mainstream network behavior modeling methods are based on neural networks, with research focusing on the optimization of features [1][2], model architectures [3][4][5], and data distribution [6][7][8]. These approaches continue to show some limits even if they have produced encouraging results in their respective fields.

First, most methods focus on extracting packet or single-flow features at the feature level, ignoring cross-flow relational features, which restricts their capacity to be generalized. For instance, Shafiq M proposes a novel feature selection algorithm that integrates correlation and area under the curve (AUC) metrics to ensure the selected feature set retains sufficient

*Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. 24, December 16-18, 2025, Sapporo, Japan.

†Corresponding Author

information[1]. Secondly, in terms of model architecture, Yao R proposes a fusion detection system of the fusion of the neural neural network (CNN) and the long-short-term memory (LSTM)[5]. The system employs CNN and LSTM to extract global and periodic features from network data, respectively. However, existing model structures are typically designed for specific tasks. Consequently, for facing different network behavior analysis tasks, it is often necessary to adjust architectural parameters such as the number of layers and module combinations, which leads to suboptimal generalizability. Many studies aim to optimize dataset distribution to improve model performance. For instance, Liu Y proposed a hybrid sampling method based on KNN and SMOTE to sample class imbalanced in flow data, with the objective of balancing the distribution of flow categories within the dataset[6]. However, the effectiveness of such approaches often relies on elaborate sample preprocessing procedures.

In recent years, large language models (LLMs) such as DeepSeek [9], GPT [10], and LLaMA [11] have achieved remarkable success in non-textual tasks across various domains, including image processing [12], time series analysis [13][14], biology [15], and web-related applications [16]. However, challenges persist in terms of task adaptation and training costs. Overall, the following challenges currently exist in the field of network behavior analysis:

- Existing neural network model architectures are often designed for specific tasks and are typically capable of analyzing only a single type of network flow, which limits their transferability and reusability across different tasks.
- The majority of existing approaches focus on extracting packet-level or single-flow features, while neglect cross-flow relationships. This oversight hinders the model's ability to recognize flow, particularly in the context of unseen flow types, leading to poor generalization performance.
- Network flow data and LLM inputs differ in modality and structure, and fine-tuning all parameters of the pre-trained model during training incurs significant costs.

This paper primarily accomplished the following:

- A network behavior analysis model architecture based on the self-attention mechanism was constructed, capable of supporting DDoS attack detection, identification of malicious encrypted flow, and fine-grained analysis of attack types, thereby enhancing the model's capability to handle multiple tasks.
- We propose a joint modeling method for flows and flow sequences. The flow embedding module is employed to represent single-flow features, while the flow sequence structure is introduced to enhance the model's ability to capture cross-flow relationships, thereby improving generalization capability.
- We implement a mapping from flow vectors to the LLM input space, adjust the LLM backbone to better accommodate flow context modeling, and freeze part of the LLM parameters to reduce training costs.

2 Scheme

The proposed model aims to achieve two main objectives: to construct an effective input structure so that network flow data can effectively adapt to the input requirements of LLM and enable LLM to extract feature information more effectively; to design a system architecture

with strong expressiveness to fully capturing the intra-flow features and inter-flow relationships of flows. To this end, three modules are designed: the flow sequence generation module, the flow embedding module, and the LLM module. The overall model architecture is illustrated in Fig. 1.

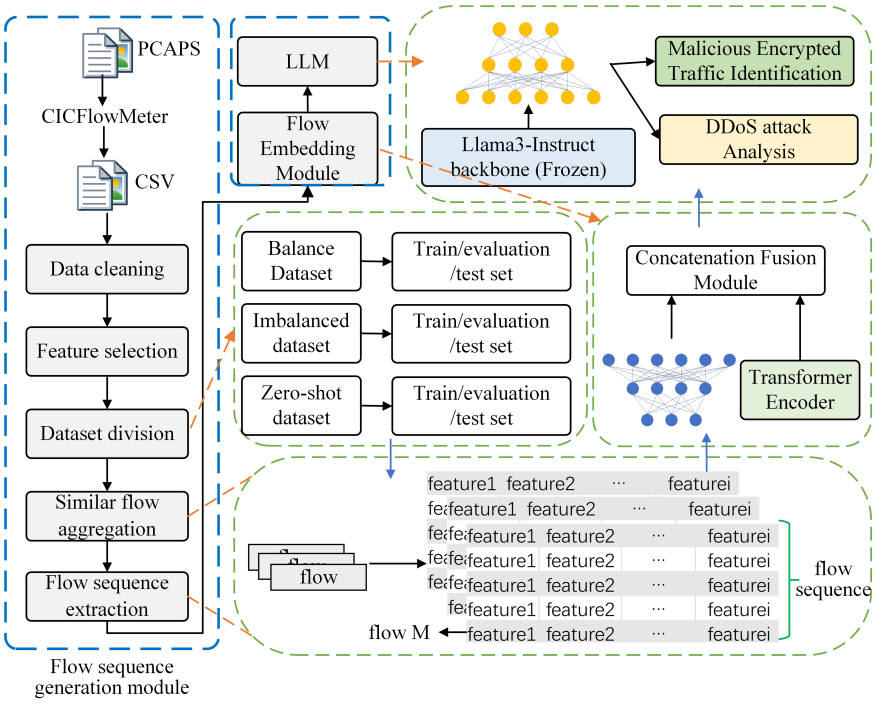


Figure 1: The Model Architecture

Initially, the device gathers raw flow data and executes preliminary data processing operations. These include constructing flows from packets, extracting features, cleaning the data, selecting relevant features, and splitting the dataset. Then, the preprocessed flow data is subjected to flows aggregation and sequence extraction to build flow sequences with contextual relationships. Subsequently, the flow sequences are fed into the dual-branch flow embedding module proposed, which performs intra-flow feature extraction and maps the flow data to the high-dimensional LLM input space. The embedded representations are subsequently input into the structurally adjusted LLM module to complete the final classification.

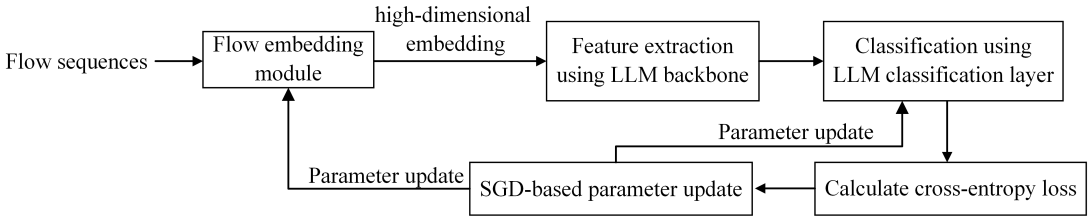


Figure 2: Joint Training Process Diagram

During training, the flow embedding module and the LLM module are jointly trained, as shown in Fig. 2, In this mechanism, the parameters of the two modules are updated concurrently. The training process employs cross-entropy loss and the Stochastic Gradient Descent (SGD) optimizer to optimize the overall model. In an effort to reduce training costs, the primary parameters of the LLM are frozen, only the flow embedding module and LLM classification layer parameters are trained.

2.1 Flow Representation Form

Common representations of network flow include packet-level, flow-level, and flow-sequence-level formats. The most granular representation is the packet, which acts as the fundamental unit for data transmission between hosts. Each packet contains attributes such as direction, protocol, timestamp, and length. A set of logically related packets can be aggregated into a flow based on the five-tuple (source IP, source port, destination IP, destination port, and protocol), along with constraints on time and TCP flags. Flows reflect interactions between packets and allow the extraction of statistical features in both upstream and downstream directions. Flow sequences are then formed by combining multiple flows. By applying specific combination strategies, the model can capture behavioral patterns across time and between flows, thus enhancing the model's capacity to represent complex behavioral patterns. In this paper, flow sequences are used as the primary representation of flows. The specific construction process is detailed in the following sections.

Flow sequences are constructed with the objective of imposing structure on unordered flows in complex networks, thereby creating contextual associations analogous to the syntax of text sequences. In the implementation, this study calculates the cosine similarity between flows based on the selected feature vectors from the DDoS dataset. The results show that flows belonging to the same attack type exhibit high similarity: 90.2% for Application DDoS attacks, 87.2% for DrDoS, 98.8% for Botnet DDoS, 93.1% for Slow DDoS, and 99.4% for Network DDoS, while benign flows show only 60.2% similarity. The construction process can be subdivided into three phases: data preprocessing, similar flow aggregation, and flow sequence extraction. The specific implementation is illustrated in in Fig. 3.

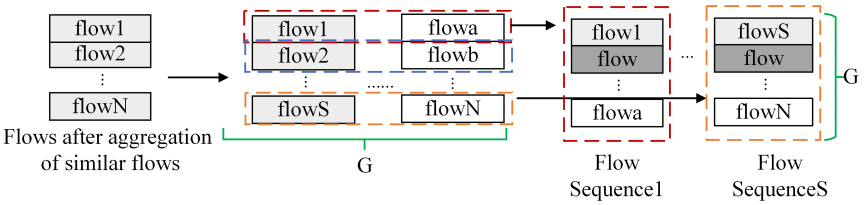


Figure 3: Flow Sequence Generation Diagram

Step 1: Data Preprocessing

Raw network flow data is collected within a specified time window using Tcpdump, and statistical features are extracted using CICFlowMeter from each flow to construct the initial flow representation vector. The data is then cleaned and standardized using Min-Max normalization to eliminate the influence of feature scale. Feature selection is performed by calculating feature importance scores using the Random Forest algorithm, and a Pearson correlation threshold of 0.9 is applied in the heatmap to filter redundant features. Subsequently, three data set evaluation scenarios were constructed based on flow type and category proportions: balanced,

unbalanced, and zero-sample scenarios. These scenarios facilitate a comprehensive evaluation of the model.

Step 2: Similar Flow Aggregation

Due to the sensitivity of LLM to input order and contextual structure, we have introduced a strategy for aggregating similar flows based on attributes. Specifically, flows is sorted based on the following attribute features in order: source port, destination port, protocol, total packet length of forward/reverse packets, and average packet length of forward/reverse packets. These attributes reflect both the connection relationship and protocol types between communicating parties, as well as the scale and direction of flows behavior at the data layer. As a result, they highlight behavioral differences among various types of network flow from the perspective of communication patterns. This step ensures that flows with similar attributes are positioned adjacently, laying the groundwork for constructing flow sequences with syntactic similarity in later stages.

Step 3: Flow Sequence Extraction

This step is crucial for constructing serialized structures, with the objective of generating flow sequences characterized by contextual structure and "syntactic" similarity. These sequences serve as the basic input units for the LLM. This process involves two phases: uniform grouping of flows and cross-group sequence extraction.

The flows is first divided into a predefined number of groups, denoted as G, which is a configurable parameter. In this study, G is set to 32. Given a total of N flows, the maximum number of flows assigned to each group is calculated as:

$$S = \left\lceil \frac{N}{G} \right\rceil, \tag{1}$$

where $\lceil \cdot \rceil$ denotes the ceiling operation, ensuring that S is rounded up to the nearest integer. The flows obtained from step 2 are sequentially filled into each group. Considering that N may not be divisible by G, the last group may contain fewer flows than the others. To address this issue, a padding strategy is applied. If the final group contains fewer than S flows, the last flow is duplicated until the group reaches the target size. This ensures that all groups contain the same number of flows. This strategy not only maintains uniform group sizes but also leverages flow similarity from step 2 to preserve structural consistency in the data.

After grouping, cross-group sequence extraction is performed. Specifically, an index $j \in [1, S]$ is to be fixed, the flow sample with index j is to be selected from each group, and the jth flow sequence F_j is to be formed:

$$F_j = \{f_{1,j}, f_{2,j}, \ldots, f_{G,j}\}, \quad F_j \in \mathbb{R}^{G \times d_{in}}, \tag{2}$$

Where $F_{i,j}$ denotes the jth flow in the ith group, and d_{in} represents the feature dimension of each flow.

The flow sequences extracted in this way contain flows with similar attributes at corresponding positions, thereby establishing a "syntax" structure similar to language. This serves to augment the LLM's capacity to capture both analogous and disparate information across flows.

2.2 Dual-branch Flow Embedding

The flow embedding module is designed to map each flow sample into a high dimensional vector. This module functions similarly to word embedding in natural language processing, enabling the LLM to perform unified downstream processing.

As shown in Fig. 4, the module is composed of two parallel branches: the MLP branch and the Transformer Encoder branch. The MLP branch performs nonlinear combination and projection on the features of each flow to enhance its feature representation. The Transformer Encoder branch learns global dependencies among features in a single flow and uncovering the interactive relationships among them. The two branches independently extract and map features, which are then concatenated and linearly fused to produce the final high-dimensional flow embedding.

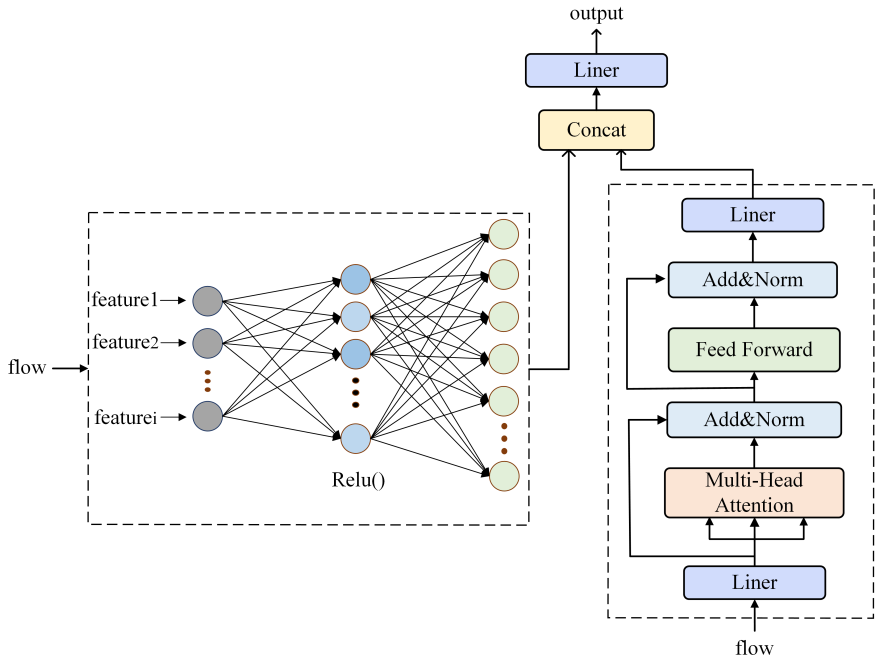


Figure 4: Flow Embedding Module Design Diagram

Let the statistical features of a flow sample be represented as $f \in \mathbb{R}^{d_{in}}$, where d_{in} denotes the number of input features.

The MLP branch processes the flow samples through a linear transformation followed by a ReLU activation to obtain its representation: $f_{\text{embed1}} = W_2 \cdot \text{ReLU}(W_1 f + b_1) + b_2$, where the hidden space dimension is set to d_{hidden1}, $W_1 \in \mathbb{R}^{d_{\text{hidden1}} \times d_{\text{in}}}$ and $W_2 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{hidden1}}}$ represent the weight matrices for the two layers, and $b_1 \in \mathbb{R}^{d_{\text{hidden1}}}$ and $b_2 \in \mathbb{R}^{d_{\text{out}}}$ are the corresponding bias terms. The output dimension d_{out} represents the embedding size of the flow representation, i.e., the token vector dimension used for input to the LLM.

To enhance the modeling of global dependencies among flow features, a Transformer encoder branch is introduced. First, the flow sample is passed through a linear mapping followed by a ReLU activation to obtain: $f_1 = \text{ReLU}(W_3 f + b_3)$, where $W_3 \in \mathbb{R}^{d_{\text{hidden2}} \times d_{\text{in}}}$ and $b_3 \in \mathbb{R}^{d_{\text{hidden2}}}$ are the weight matrix and bias term of this layer, respectively. The output is then fed into a standard Transformer encoder to obtain: $f_2 = \text{TransformerEncoder}(f_1)$, where TransformerEncoder denotes an encoder layer consisting of a self-attention mechanism and a feedforward network. Finally, a linear layer is applied to project the output to the same embedding dimension as the MLP branch: $f_{\text{embed2}} = W_4 f_2 + b_4$, where $W_4 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{hidden3}}}$ and $b_4 \in \mathbb{R}^{d_{\text{out}}}$ are the weight

matrix and bias term of this layer.

The outputs of the two branches are concatenated along the feature dimension, then reduced to the d_{out} dimension through a linear layer, yielding the final embedding representation: $f_{\text{embed}} = W_5(f_{\text{embed1}} \parallel f_{\text{embed2}}) + b_5$, where $W_5 \in \mathbb{R}^{d_{\text{out}} \times d_{2\text{out}}}$ and $b_5 \in \mathbb{R}^{d_{\text{out}}}$ are the weight matrix and bias term of this layer. Ultimately, f_{embed} represents the embedding of each flow in high-dimensional space, and a sequence of multiple flows can be directly input into subsequent modules.

2.3 LLM Restructuring

To better adapt the model for network behavior analysis tasks, the open-source large model Llama3-Instruct is used as the foundation and makes structural adjustments in accordance with the characteristics of the task. The primary focus of these adjustments is the refinement of the self-attention mechanism and the output head, aiming to address the limitations of the original model designed for language modeling.

1. Adjustment of the self-attention mechanism

The original pre-trained model utilizes a mask-based unidirectional self-attention mechanism, which restricts each token to attending only to its preceding tokens. Though this approach is suitable for language generation tasks, it limits the modeling of global dependencies between flows. To address this, we reset the mask matrix and remove directional constraints on attention, allowing the model to fully capture similarities and differences between flows. This allows for the extraction of richer flow representations and the improvement of network behavior analysis capabilities.

The causal attention mask is defined as:

$$M_{ij} = \begin{cases} 0, & j \leq i, \\ -\infty, & j > i. \end{cases} \tag{3}$$

We set the attention mask matrix to all zeros, enabling information interaction between all positions:

$$M_{ij} = 0, \quad \forall i, j \tag{4}$$

Resetting the mask removes directional restrictions during attention computation. Combined with softmax normalization, the model can assign dynamic association weights between any two flows in the sequence:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right)V, \tag{5}$$

Where $Q, K, V \in \mathbb{R}^{g \times d_k}$ denote the query, key, and value matrices, respectively. d_k is the dimension of each attention head, M is the adjusted mask matrix, and g is the length of the flow sequence.

Unlike convolutional or recurrent networks, which can only model local or sequential dependencies, the bidirectional self-attention mechanism can capture global relationships and extract potential information about the relationship between any two flows.

2. Output Head Reconstruction

The original LLM uses a standard language modeling head that predicts the next-token distribution.Its output is shown as:

$$\hat{y}_{\text{token}} = \text{softmax}(Wx_t + b), \tag{6}$$

where $x_t = [x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(d_{\text{model}})}] \in \mathbb{R}^{d_{\text{model}}}$ denotes the representation vector of the current token, $W \in \mathbb{R}^{d_{\text{model}} \times V}$ and $b \in \mathbb{R}^V$ are the learnable weight matrix and bias term, respectively. Here, V is the vocabulary size.

In the flow detection scenario, the model is required to classify each flow. Therefore, we introduce an output head for classification tasks to replace the original language modeling output structure. The classification output layer consists of two fully connected layers, and the output is the category probabilities corresponding to the different network behavior analysis tasks:

$$\hat{y} = \text{softmax}(W_7(W_6 x + b_6) + b_7), \tag{7}$$

where $x = [x^{(1)}, x^{(2)}, \ldots, x^{(d_{\text{model}})}] \in \mathbb{R}^{d_{\text{model}}}$ is the final vector representation of each flow from the LLM's backbone, $W_6 \in \mathbb{R}^{d_{\text{hidden4}} \times d_{\text{model}}}$ and $W_7 \in \mathbb{R}^{C \times d_{\text{hidden4}}}$ are the weight matrices of the two layers, and $b_6 \in \mathbb{R}^{d_{\text{hidden4}}}$, $b_7 \in \mathbb{R}^C$ are the corresponding bias terms. d_{hidden4} is the dimension of the intermediate hidden layer, and C is the total number of classes. The output $\hat{y} \in \mathbb{R}^C$ represents the predicted probability distribution over all classes.

To optimize classification performance, the cross-entropy loss is used to compute the difference between the predicted results and the true labels.

$$\text{loss} = -\sum_{i=1}^{C} y_i \log \hat{y}_i, \tag{8}$$

where $\hat{y}_i$ denotes the predicted probability for class i, and y_i is the one-hot encoded ground truth label.

3 Evaluation

3.1 Experiment Settings And Datasets

The experiments are conducted under the following hardware and software: Ubuntu 20.04 operating system; a 40-core Intel®Xeon® Silver 4210R processor; and four NVIDIA GeForce RTX 3070 graphics cards. The study used CUDA version 12.1. The development language was Python 3.8, and deep learning modeling was based on PyTorch 2.3.0 (cu121). The open-source large language model Llama3-Instruct-8B is adopted as the backbone architecture, with structural adjustments made to suit network behavior analysis tasks. In this study, the flow sequence length is set to 32, the batch size is set to 16, the learning rate is set to 0.001, and the Adam optimizer is employed.

This study focuses on DDoS attack detection and malicious encrypted flow identification as the main experimental scenarios. DDoS attacks overwhelm target systems with massive flow requests, while malicious encrypted flow uses encryption protocols to hide its intentions. The datasets used in this study are derived from three sources. First are the DDoS attack and normal flow data collected in a laboratory environment. These data cover five major categories of attack types, each of which encompasses multiple subtypes. Second is the publicly available CICDDoS2019 dataset. Third are the encrypted flow datasets, which include normal and various encrypted flow samples generated by malicious software. Data from all categories in each scenario are randomly divided into training and testing sets at a ratio of 8:2. Each dataset is utilized for distinct experimental tasks, as detailed below:

(1) Analysis of the Five Major Categories of DDoS Attacks (Laboratory Data): Includes six categories: Benign, SlowDDoS, Network, Application, DrDoS, and Botnet. The test set contains 6,000 benign flows and 6,000 malicious flows;

(2) Specific Types of DDoS Attacks (Laboratory Data): Includes Benign and 22 specific attack types: Shrew, SlowHeaders, SlowReader, SlowBody, ACK-Flood, UDP-Flood, CC, HTTP-Flood, HTTP-Get, HTTP-Post, SYN, Memcached, Chargen, SSDP, NTP, SNMP, Ares, BYOB, Mirai, Zeus, and IRC-Botnet. The test set contains 10,000 benign flows and 2,000 malicious flows to simulate an imbalanced traffic scenario;

(3) CICDDoS2019 Attack Detection: Includes Benign and DDoS categories. The test set contains 6,000 benign flows and 6,000 malicious flows;

(4) CICDDoS2019 zero-sample analysis: The training set includes Benign and some DDoS attack types (SYN, NTP, DNS), while the test set uses attack types not seen during training (SNMP, LDAP, UDP, MSSQL, PortMap, etc.);

(5) Malicious encrypted flow detection: In the detection scenario, Benign and Malicious labels are set. The test set contains 6,000 benign flows and 6,000 malicious flows.

For the DDoS attack dataset, feature selection was performed to retain the most relevant attributes. The selected features include: 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. For the malicious encrypted traffic dataset, the following features were selected: Start Time, TCP Out Flags, Byte Distribution Mean, End Time, Outbound Inter-Packet Time Mean, Flow Expire Type, Dst Port, Inbound Inter-Packet Time Mean, TCP Inbound Flags, TCP Inbound MSS Option, TCP Inbound Option Length, Byte Distribution Standard Deviation, Outbound IP TTL, TCP Outbound Option Length, Outbound Bytes, First TCP Sequence Number, TCP Outbound First Window Size, Payload Bytes, TCP Inbound Window Scale Option, Entropy and Total Entropy.

3.2 Evaluation indicators And Comparison settings

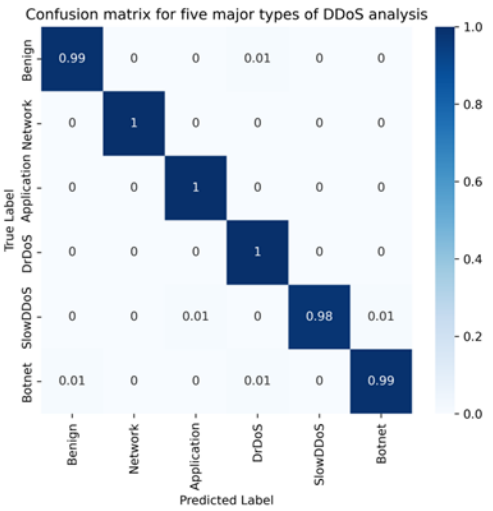
To comprehensively evaluate the performance of the proposed model, four commonly used evaluation metrics are employed: accuracy, precision, recall, and F1-score. In the comparative experiments, three widely adopted models are used as baselines: Random Forest (RF), CNN and LSTM. These comparisons aim to verify the effectiveness and demonstrate the superiority of the proposed model across various network behavior analysis tasks.

3.3 DDoS Attacks Detect Result

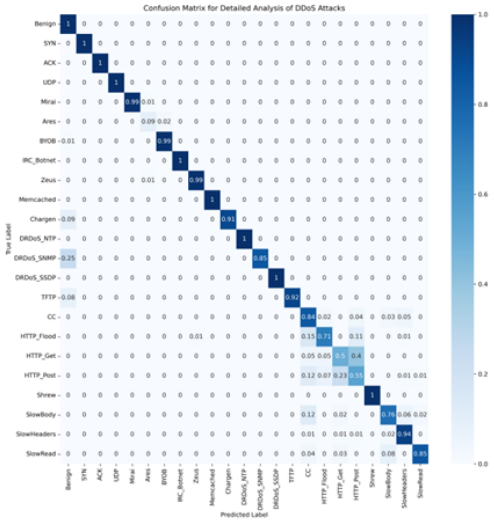
DDoS attack detection experiments are conducted using two data sources: a laboratory-collected dataset and the publicly available CICDDoS2019 dataset. Tasks (1) to (4), as defined in Section 3.1, are performed in this setting. The detection results are presented in the form of a confusion matrix in Fig. 5(a)–(d). The results of the comparison experiments with other models are shown in Table 1.

As shown in Table 1, the system model proposed in this study performs well in multiple DDoS attack detection experimental scenarios. It demonstrates superior detection performance with respect to accuracy, precision, recall, and F1-score, particularly in analyzing 23 specific types and performing zero-shot analysis. These results demonstrate the model's effectiveness and generalization ability in network behavior analysis tasks.

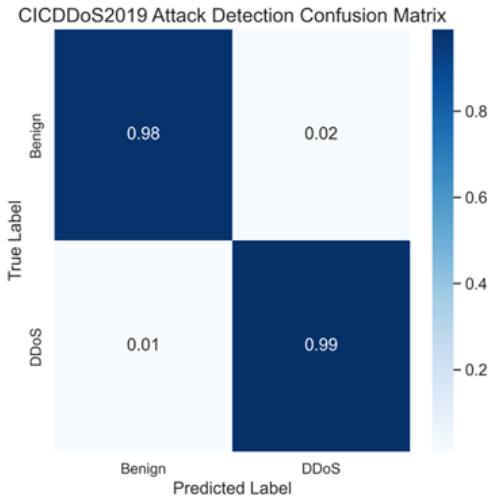
To verify the effectiveness of the proposed flow sequence structure, two groups of ablation experiments were conducted: one using single-flow detection and the other using time window–



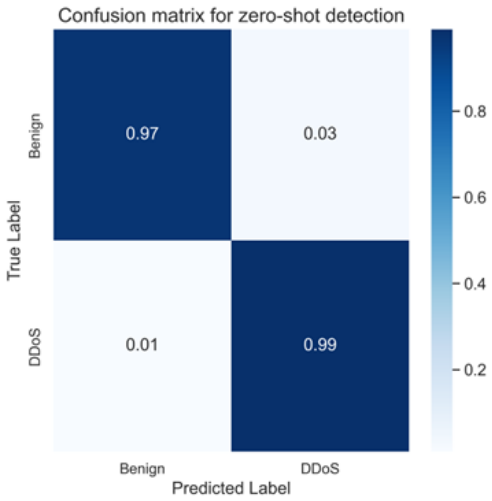
(a) Five major types of DDoS attacks analysis



(b) Detailed analysis of DDoS attacks



(c) CICDDoS2019 attack detection



(d) CICDDoS2019 zero-shot detection

Figure 5: DDoS Attacks Detect Result

based flow sequence detection. Under the same zero-shot detection dataset and unified model configuration, the detection accuracies of the two methods were 96.37% and 93.99%, respectively, which are 2.13% and 4.51% lower than the proposed similarity-based flow aggregation approach. These results demonstrate that the proposed method can more effectively capture the contextual dependencies among flows.

Model	Evaluation	DDoS Detection	Five Types Analysis	Detailed 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
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	0.9700	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

Table 1: DDoS Attacks Detect Result

3.4 Encrypted Malicious Flow Detect Result

This section presents experiments on detecting malicious encrypted flow. These experiments were performed using a dataset of malicious encrypted flow collected in a laboratory environment. These experiments follow the procedure outlined in Section 3.1. The detection performance, along with comparisons against baseline models, is summarized in Table 2.

Scene	Evaluation	Our	RF	CNN	LSTM
Malicious Encrypted Flow Detection	Accuracy	0.9976	0.9870	0.9760	0.9755
	Precision	0.9987	0.9861	0.9964	0.9517
	Recall	0.9983	0.9731	0.9729	0.9798
	F1 Score	0.9985	0.9794	0.9845	0.9648

Table 2: Malicious Encrypted Flow Detect Result

As illustrated in Table 2, the proposed model achieves better performance compared to the baseline methods in detecting malicious encrypted flow.

3.5 Result Analyse

The advantages demonstrated in Sections 3.3 and 3.4 can be attributed to the joint modeling mechanism of flows and flow sequences. This mechanism overcomes the limitations of relying solely on single-flow or packet-level features. The model can capture potential associations between flows by constructing flow sequence structures, incorporating contextual information from other flows, and leveraging the modeling capabilities of LLMs. This enhances its ability to identify complex flow behaviors, particularly in multitype attack analysis and zero-sample

detection tasks. In the fine-grained analysis of 23 specific DDoS attack types, the model maintains high accuracy and demonstrates good recall capability. In zero-shot scenarios, the model's detection capability significantly outperforms other models, indicating that it can accurately identify most attack types and possesses strong generalization capabilities. Additionally, the model consistently outperforms traditional neural network models in identifying malicious encrypted flow, further substantiating the effectiveness of the proposed approach.

4 Conclusion

Addresses the limitations of conventional neural network models in network behavior analysis, particularly their task-specific design and lack of generalization, this paper proposes a network behavior analysis architecture based on LLMs. The architecture achieves effective mapping of network flow features into a high-dimensional semantic space by constructing flow sequence with contextual relationships and designing a dual-branch flow embedding module. Building on this architecture, the self-attention mechanism and output structure of the pre-trained LLM are further adapted to better suit network flow analysis. Experimental results show that the proposed model achieves outstanding performance across multiple evaluation metrics, including accuracy, precision, recall, and F1 score.

5 Acknowledgments

This paper is supported by National Key R&D Program of China under Grant No.2018YFA0701 604 and NSFC under Grant No.62341102.

References

[1] Shafiq M, Tian Z, Bashir A K, et al. CorrAUC: A malicious bot-IoT traffic detection method in IoT network using machine-learning techniques. IEEE Internet of Things Journal, 2020, 8(5): 3242-3254.

[2] Sun X, Yuan Z, Kong X, et al. Communication-efficient and privacy-preserving aggregation in federated learning with adaptability. IEEE Internet of Things Journal, 2024, 11(15): 26430-26443.

[3] Wang Z, Liu Y, He D, et al. Intrusion detection methods based on integrated deep learning model. computers & security, 2021, 103: 102177.

[4] Xiao Y , Xiao X .An Intrusion Detection System Based on a Simplified Residual Network.Information (Switzerland), 2019, 10(11):356.

[5] Yao R , Wang N , Liu Z ,et al.Intrusion Detection System in the Advanced Metering Infrastructure: A Cross-Layer Feature-Fusion CNN-LSTM-Based Approach.Sensors, 2021, 21(2):626.

[6] Liu Y, Wu L. Intrusion detection model based on improved transformer. Applied Sciences, 2023, 13(10): 6251.

[7] Vu L, Bui C T, Nguyen Q U. A deep learning based method for handling imbalanced problem in network traffic classification//Proceedings of the 8th international symposium on information and communication technology. 2017: 333-339.

[8] Ullah, Farhan, Shamsher Ullah, Gautam Srivastava and Chun-Wei Lin. "IDS-INT: Intrusion detection system using transformer-based transfer learning for imbalanced network traffic." Digit. Commun. Networks 10 (2023): 190-204.

[9] Liu A, Feng B, Xue B, et al. Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437, 2024.

[10] Achiam J, Adler S, Agarwal S, et al. Gpt-4 technical report. arXiv preprint arXiv:2303.08774, 2023.

[11] Touvron H, Lavril T, Izacard G, et al. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023.

[12] Wang W, Chen Z, Chen X, et al. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. Advances in Neural Information Processing Systems, 2023, 36: 61501-61513.

[13] Lu K, Grover A, Abbeel P, et al. Frozen pretrained transformers as universal computation engines. In: Proceedings of the AAAI conference on artificial intelligence. 2022, 36(7): 7628-7636.

[14] Liu Y, Qin G, Huang X, et al. Autotimes: Autoregressive time series forecasters via large language models. Advances in Neural Information Processing Systems, 2024, 37: 122154-122184.

[15] Li Y, Li Z, Zhang K, et al. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. Cureus, 2023, 15(6).

[16] Lin X, Xiong G, Gou G, et al. Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification//Proceedings of the ACM Web Conference 2022. 2022: 633-642.