Cybersecurity

RESEARCH

Open Access

Enmob: Unveil the Behavior with Multi-flow Analysis of Encrypted App Traffic

Ge Mengmeng[1], Feng Ruitao[3,4], Liu Likun[1], Yu Xiangzhan[1*], Sachidananda Vinay[2], Xie Xiaofei[3] and Liu Yang[2]

Abstract

In the contemporary digital landscape, mobile applications have become the predominant conduit for internet connectivity and daily tasks. Simultaneously, the advent of application encryption technology has safeguarded users' privacy. However, this encryption, while fortifying privacy, introduces challenges to security by hindering the effective management of network applications within encrypted data streams. Conventional detection methods for encrypted application traffic, relying heavily on statistical metrics like payload, packet size, and distribution, are constrained to single traffic flows, often yielding results of limited specificity. To address this limitation, our paper introduces an innovative approach that elucidates the multi-flow nature of application behavior traffic and provides context to encrypted application traffic. This method offers a more nuanced and comprehensive perspective for understanding and representing network traffic, even when encrypted. The efficacy of our approach was evaluated using a substantial volume of real network traffic data. Results indicate that our method achieves an average accuracy of 0.958 in identifying application behavior traffic and 0.955 in classifying application traffic. These outcomes signify a substantial enhancement over single network flow-based detection methods, demonstrating a notable 5.3% improvement.

Keywords Traffic analysis, Encryption traffic, Behavior Traffic Classification

Introduction

In the ever-evolving landscape of network technology, the analysis of traffic patterns plays a pivotal role in both network management and security. Such as encryption type identification (Wang et al. 2017a), IoT device identification [2], intrusion detection (Borders et al. 2012), anomaly detection (Mirsky et al. 2018) and network service type identification (Draper-Gil et al. 2016), etc. The classification of application behavior and application traffic, integral components of traffic analysis, has gained significant attention in recent years. Effective network management and robust network security are heavily reliant on the ability to accurately interpret and classify network traffic.

Prior research in network analysis predominantly relied on pattern-based methodologies (Bujlow et al. 2015). However, the swift evolution of encryption technology has rendered these pattern-based approaches ineffective. Consequently, the adoption of learning-based network intrusion detection systems has garnered significant attention within both academic (Li et al. 2018; Holland et al. 2021) and industrial (Mirsky et al. 2018) domains. Learning-based strategies encompass both machine learning-based (Shafiq et al. 2016), and deep learning-based (Wang et al. 2017b). Within the realm of machine learning-based approaches, statistical feature methods are prevalent, including the utilization of algorithms such as random forest (Shang et al. 2019), SVM (Feng et al. 2019), and KNN (Xie et al. 2012). Detection models based on spatial distribution, such as sequential models (Barradas et al. 2021), convolutional neural networks (Sirinam

*Correspondence:
Yu Xiangzhan
yzx@hit.edu.cn
[1] School of Cyberspace Science, Harbin Institute of Technology, 92 West Dazhi St, Harbin 150001, Heilongjiang, China
[2] School of Computer Science and Engineering, Nanyang Technlogical University, 50 Nanyang Ave, Singapore, 639798 Singapore, Singapore
[3] School of Computing and Information Systems, Singapore Management University, 81 Victoria St, Singapore, 188065 Singapore, Singapore
[4] Faculty of Science and Engineering, Southern Cross University, Gold Coast Airport, Terminal Dr, Bilinga, QLD 4225, Australia

Springer Open

© The Author(s) 2025. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

et al. 2018), and recurrent neural networks (Liu et al. 2019), have also been widely employed in this context.

Traditional approaches in traffic analysis have primarily focused on the examination of single network traffic flows. While these methods have been somewhat effective, they fall short in capturing the intricacies of applications that generate multiple network flows. This limitation poses a significant challenge in accurately classifying and understanding application behaviors and traffic patterns.

Surprisingly, multiple flows in traffic analysis have not been systematically studied, and little research has been carried out on the flow relationship between application behaviors and multiple flows and the context of application behaviors traffic within an application. To make up for the above gap, this paper proposes three aims. Firstly, explore and define the relationships between different network flows in application behavior traffic with multi-flow. Secondly, how to build a model to describe the application behavior traffic with multiple flows. Thirdly, explore the contextual relationship of different application behavior traffic within the application and build a model to identify different application traffic.

In order to achieve the above goals, through a large-scale case study, we first define the relationship between multiple flows from the aspects of attribute characteristics and the timing sequence of network flows. Then, we propose a novel network flow graph model to represent the relationship between multiple network flows. Finally, we build an application traffic classification model based on multi-flow behavior traffic by analyzing the context of application behavior traffic within the application. Those approaches represent a significant departure from traditional single-flow analysis, offering a more nuanced and comprehensive understanding of network traffic.

This paper aims to explore novel approaches to model application behavior traffic through the lens of multi-flow traffic analysis. It seeks to bridge the gap in current methodologies that fail to consider the collective features of multi-flow traffic. The exploration extends to how these relationships can be effectively utilized to model application traffic, considering the context of application behavior traffic. Our contributions are as follows:

- We initially introduced the concept of multi-flow relationships within application behavior traffic. Through extensive analysis, we defined three distinct types of network multi-flow relationships, offering a novel and insightful perspective in this domain.
- We proposed an innovative multi-flow graph model, which is a pioneering approach to represent and analyze multiple network flows, enabling more effective classification of application behavior traffic based on multi-flow traffic similarity.
- We proposed a new application traffic classification method that uniquely integrates the multi-flow characteristics of application behavior traffic with the context of behavioral traffic. This method not only enhances traffic classification accuracy but also underscores the significance of traffic analysis in collaborative detection across network flows.

The organization of this paper is as follows: Sect. introduces the problem's background and current challenges. Section delves into related research, providing a theoretical backdrop. Section and Sect. are pivotal, introducing the construction of multi-flow relationship and behavioral context models. Section focuses on evaluating the accuracy and performance of these models. Section engages in a discussion on the findings, acknowledging the limitations of the study. Finally, Sect. concludes the paper with a summary, encapsulating the key insights and contributions of the research.

Related work

Network Traffic Analysis

Network traffic analysis receives extensive attention from network security researchers, and is used in various tasks of network security. Traffic analysis can infer user's actions (Conti et al. 2016) in android platforms and the web page user view (Shen et al. 2021). Applications identification help the manager clearly know applications used within the organization by traffic analysis (Shen et al. 2021; Shapira and Shavitt 2021). Traffic using different encryption technologies can hide transferred data, hence, traffic encryption technologies detection (Wang et al. 2017a) and traffic categories(Shapira and Shavitt 2021) (e.g. VoIP, viedo, etc.) detection in encryption channels are used in network management.

Besides, network intrusion detect system (Borders et al. 2012) detects network intrusion behavior by analyzing network traffic, such as, bot command and control connections detection (Jacob et al. 2011), tor routing attack traffic detection (Sun et al. 2015), and website injection traffic detection (Nakibly et al. 2016). Besides, traffic analysis also applied to malware traffic detection (Bartos et al. 2016; Wang et al. 2017b), abnormal traffic detection (Li

et al. 2020; Dong et al. 2021; Li et al. 2021), and website fingerprinting attack (Wang and Goldberg 2017).

Approaches of Encrypted Traffic Classification

Deep Learning-Based Methods: Over recent years, deep learning techniques have gained significant traction in encrypted traffic classification. These methods typically employ neural networks, such as Convolutional Neural Networks (CNNs) (Wang et al. 2017a) or Recurrent Neural Networks (RNNs) (Wang et al. 2019), to analyze traffic patterns and identify specific characteristics of encrypted data. Statistical Analysis: Statistical analysis remains a popular method for encrypted traffic classification. This approach involves analyzing the statistical properties of traffic flows, such as packet sizes, timing, and flow duration, to infer the type of application or service (Aceto et al. 2018; Bar-Yanai et al. 2010). Behavioral Analysis: Behavioral analysis methods focus on the behavior of applications or users rather than the content of the traffic itself. By examining patterns such as data flow rates and connection intervals, these methods aim to classify traffic based on application behavior profile (Zheng et al. 2020; Wang et al. 2019). Machine Learning with Feature Engineering: Machine learning techniques combined with feature engineering have been extensively used for encrypted traffic classification. This approach involves selecting and engineering specific features from traffic data, which are then used to train machine learning models like Support Vector Machines (SVMs) (Fan et al. 2017) or Random Forests (Liu et al. 2021).

In addition, there are semi-supervised detection methods, such as FlowPrint (Van Ede et al. 2020), which identifies applications by extracting and clustering network flow features such as periodicity and bytes transferred. Moreover, there are pre-trained fine-tuning models, like ET-Bert (Lin et al. 2022) for traffic classification, which treats encrypted traffic as sequences of "sentences" and "words," utilizing the BERT language model to capture complex patterns in packet metadata. These methods showcase advanced techniques for analyzing encrypted network traffic without decryption.

Application Behavior Traffic Similarity based on Multi-flow

In this section, we delve into the intricacies of network applications, which are characterized by multiple application behaviors. These behaviors often necessitate the cooperative integration of multi-flows. For instance, an application behavior like sending pictures typically involves the concerted action of three distinct network flows, as shown in Fig. 1. Our focus here is to analyze multi-flow relationships within application behavior traffic, as well as the contextual interplay of various multi-flow application behaviors inside the application.

Multi-flow Contruction based on Graph

As shown in Fig. 2, we utilize a connected graph representation to represent network traffic. The nodes are the traffic flow, and the edge represents the relationship between different flows.

For network trafic flow set $F = f_1, f_2, \ldots, f_i, \ldots$, which is generated in an application behavior. The relationships between traffic flow can be divided into three types illustrated as follows:

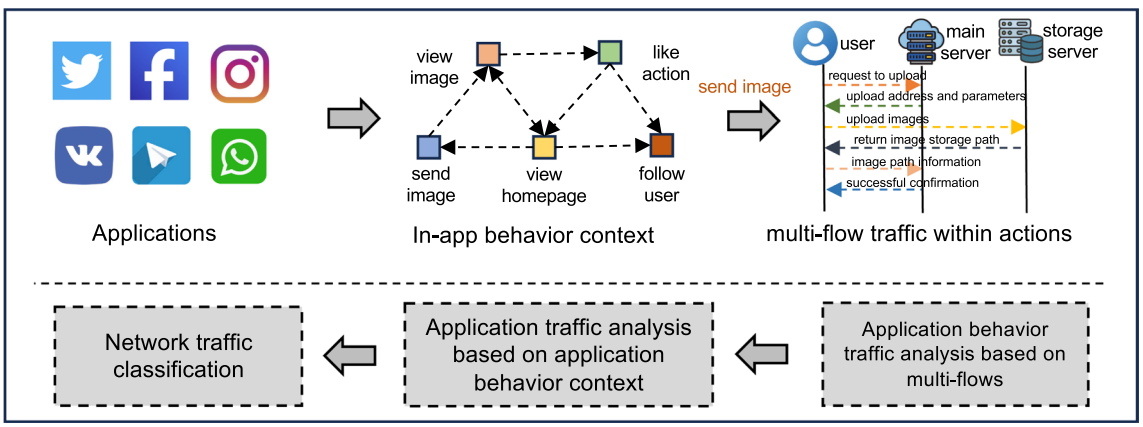


Fig. 1 Application behavior context and multi-flow relationships within application behavior

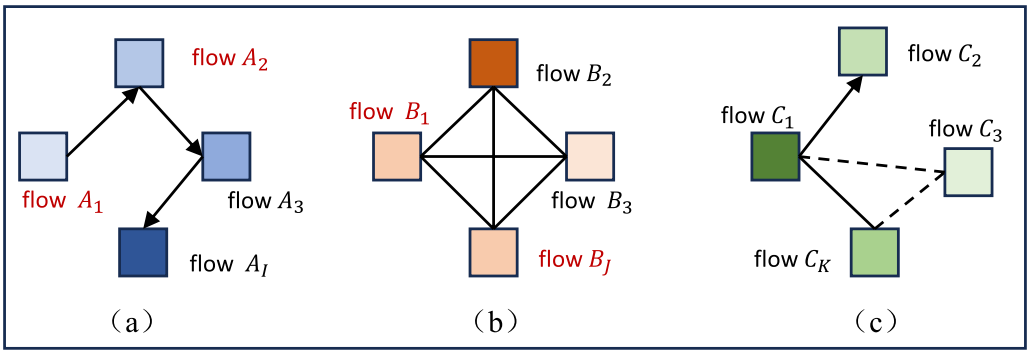


Fig. 2 Different relationships of network flows: **a** Adjoint, **b** Partial order, and **c** Independent

Adjoint: flow f_i and flow f_j, they always appear together in the traffic of application behavior, hence, we mark it as $f_i \sim f_j$. The adjoint relationship satisfies reflexivity and symmetry. For any flow f_i and f_j, if there exists $f_i \sim f_j$, there must be $f_j \sim f_i$.

Partial Order: flow f_i and flow f_j appear together, and flow f_i alway appear early than flow f_j, we mark it as $f_i \prec f_j$. The partial ordering relationship satisfies reflexivity, antisymmetry and transitivity. For any flow f_i and f_j, if there exists $f_i \prec f_j$ and $f_j \prec f_i$, there must be $f_i = f_j$. For any flow f_i, f_j and f_k, if $f_i \prec f_j$ and $f_j \prec f_k$, there must be $f_i \prec f_k$. It is worth noting that the partial ordering relationship of flows is included in the adjoint relationship of flows. Two flows that satisfy the partial ordering relationship must also satisfy the adjoint relationship at the same time.

Independent: if there is no fixed sequential relationship between flow f_i and flow f_j, in other words, the relations may not appear every time, we would define there is an independent relationship between the single flow f_i and f_j marked as $f_i \perp f_j$. The independent relationship of flows satisfies reflexivity and symmetry. For any flow f_i and f_j, if there exists $f_i \perp f_j$, there must be $f_j \perp f_i$ received. Two flows that satisfy the independent relationship must not satisfy the adjoint or partial order relationship. Similarly, two flows that satisfy the adjoint relationship or partial order relationship must not satisfy the independent relationship.

We use edges to represent the network traffic flow's relationship and nodes to represent the network traffic flow. The node stores a single flow's basic information and statistical features, including flow communication server domain name, communication server port, and traffic layer statistical characteristics, etc. Here, we use those traffic features as shown in the Table 1.

Network Traffic Processing

In our network traffic analysis methodology, we begin by segmenting the traffic based on the five-tuple classification, which includes source and destination IP addresses, source and destination ports, and the protocol used. Following this initial categorization, we further partition the network traffic into *burst* (Taylor et al. 2016), a sudden, short-lived increase in the volume of data transmission over a network caused by application behavior, based on specific parameters such as timestamps and packet sizes. Specifically, we define a burst based on the activity within a continuous threshold time window. If there is a period of inactivity, measured as a lack of packet transmissions, lasting for

Table 1 Features of each network traffic flow node

Features type	Features we used
upstream direction	#packet numbers, #stream size, #max pkt size, #min pkt size, #pkt Avg. size, #pkt Std. deviation
downstream direction	#packet numbers, #stream size, #max pkt size, #min pkt size, #pkt Avg. size, #pkt Std. deviation
total	#max pkt size, #min pkt size, #Avg. pkt size, #pkt size Std. deviation

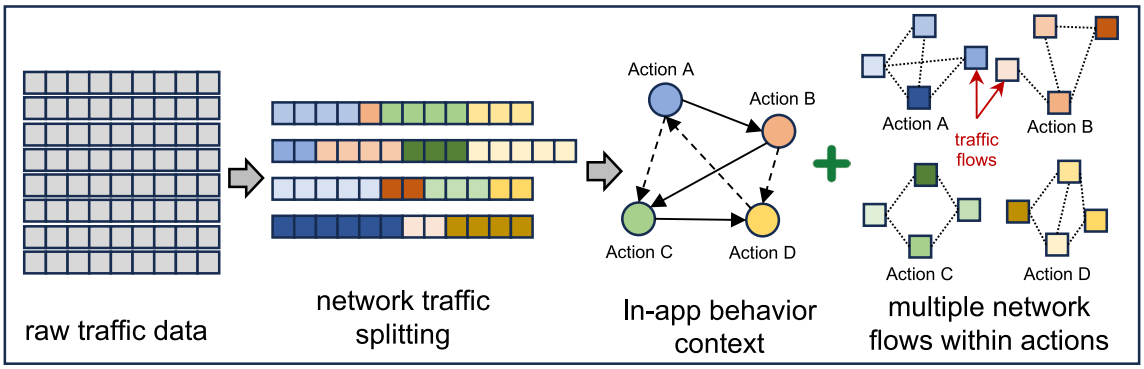


Fig. 3 Construcion of the multi-flow model within application behavior traffic

more than a *burst threshold* of 1 s, used in the reference (Taylor et al. 2016), we consider this as the boundary between two distinct bursts.

Each of these bursts is then represented as a distinct node within a network multi-flow traffic graph. This graph-based representation allows us to capture multiple snippets of network traffic, offering a granular view of the traffic flow. Subsequently, we segregate the network traffic into various application behavior traffic samples, using time intervals as the primary differentiator, as shown in Fig. 3.

Within each traffic behavior sample, we further dissect the data into multiple network flow segments. This segmentation is again based on groups defined by the five-tuple classification. Through this methodical division and analysis, we can construct a comprehensive multi-flow graph. This graph not only represents the application behavior traffic but also facilitates a deeper understanding of the underlying patterns and relationships within the network traffic.

In our approach, we group multiple flows by domain name. Utilizing domain names as the grouping criterion allows us to comprehensively represent the traffic linked to specific applications or services, which frequently span multiple network connections. Importantly, while we recognize that network traffic and user behaviors are inherently dynamic, our domain name-based grouping strategy is inherently designed to accommodate such variability. Applications and web services typically generate network flows consistently associated with particular domains, making this approach robust against fluctuations in flow specifics, such as IP addresses or ports. This constancy provides a stable foundation for our traffic classification efforts.

Multi-flow Traffic Similarity

In this paper, the method for multi-flow matching calculates the overall multi-flow similarity, denoted as $\rho(G, G')$, for a given multi-flow structure $G = (V, E)$ (where V represents nodes/single flow collections, and E represents edge collections) and the target multi-flow $G' = (V')$ (containing only the target single flow collection V', without a logical description). The specific algorithm is described in Algorithm 1.

Algorithm 1 Multi-Flow Similarity Calculation

Input: $G = (V, E)$, $G' = (V')$
Output: $\rho(GG')$

$M' \leftarrow \emptyset$;
$\rho(GG') \leftarrow 0$;
for $i \in V'$ **do**
  $I \leftarrow findMatch(i, V)$;
  if $I \neq NULL$ **then**
    $\rho_{flow}(i, I) \leftarrow calculateFlowSimilarity(i, I)$;
    $M' \leftarrow M' \cup i$;
for $i \in M', j \in M', i \neq j$ **do**
  $I \leftarrow findMatch(i, V)$;
  $J \leftarrow findMatch(j, V)$;
  $Edge_{IJ} \leftarrow getEdge(I, J, V)$;
  if $Edge_{IJ}.isPO = true$ **then**
    $POtype' \leftarrow checkPOtype(i, j)$;
    if $POtype' = Edge_{IJ}.POtype$ **then**
      $l_{flow}(i, j) \leftarrow 1$;
    else
      $l_{flow}(i, j) \leftarrow 0$;
  else
    $l_{flow}(i, j) \leftarrow Edge_{IJ}.Pso_{IJ}$;
  $\rho_{joint}(i, j) \leftarrow \rho_{flow}(i, I) * l_{flow}(i, j) * \rho_{flow}(j, J)$;
  $\rho(GG') \leftarrow \rho(GG') + \rho_{joint}(i, j)$;
$\rho(GG') \leftarrow \rho(GG') \ / \ sizeOf(E)$;

For the given target flow set $G' = (V')$, traverse the nodes i contained in it. Map them to the nodes $I \in V$ based on their basic information, such as the server domain name and server port, and calculate the single-flow similarity between them.

The single-flow similarity can be calculated in two ways: The first method utilizes the statistical characteristics of the nodes' single flows, combines them into vector form to obtain a feature vector of single flows, and then calculates the feature vector distance between flow f_i and flow f_j using the cosine similarity algorithm, represented as $\rho_{flow}(i,j)$. This approach calculates the similarity between two single flows based on their feature vectors. The second method employs the Dynamic Time Warping algorithm (DTW). We can use the length sequences of data packets for both candidate single flows as inputs of the DTW algorithm. Then, the distance on the length change curve of these two flows is calculated using DTW to describe their similarity.

Once the matching and similarity calculation for the single flow nodes are completed, we obtain a single flow matching set M' that contains nodes or single flows from V' that are matched with V. Next, we can calculate the logical similarity $\rho_{logic}(i,j)$ between graphs G and G' as follows: For the nodes i and j in M', as well as the matched nodes I and J in V, retrieve the edge $edge_{I,J}$ connecting nodes I and J from G. The edge $edge_{I,J}$ stores three items $\langle isPO, POtype, Pso_{(I,J)}\rangle$. $isPO$, a boolean value indicating whether this edge is a partial order relation; $POtype$, a description of the partial order type; thirdly, $Pso_{I,J}$, the probability that the two vertices represented by the edge connect in the flows simultaneously. If $isPO$ is true, check if nodes i and j satisfy the partial order type $POtype$. If they do, the flow logic relationship parameters $r_{flow}(i,j) = 1$; otherwise, $l_{flow}(i,j) = 0$. If $isPO$ is false, $l_{flow}(i,j) = Pso_{I,J}$. For the nodes i and j, and having already calculated their flow similarities $\rho_{flow}(i,I)$, $\rho_{flow}(j,J)$ and $l_{flow}(i,j)$, we

can further calculate their joint similarity $\rho_{joint}(i,j)$ as shown in Eq. 1:

$$\rho_{joint}(i,j) = \rho_{flow}(i,I) * l_{flow}(i,j) * \rho_{flow}(j,J) \quad (1)$$

Finally, the multi-flow similarity $\rho(G,G')$ between G and target G' is calculated as shown in Eq. 2:

$$\rho(G,G') = \frac{\sum_{i,j}^{i,j\in V'} \rho_{joint}(i,j)}{sizeOf(E)} \quad (2)$$

By calculating the multi-flow similarity $\rho(G,G')$, we can quantitatively describe the similarity between the given multi-flow structure $G=(V,E)$ and the target multi-flow $G'=(V')$. Based on the selected threshold, it can be determined whether they are successfully matched.

Application Behavior Traffic Similarity Calculation

In previous sections, the multiple flow matching degree calculation algorithm designed for multiple flow structures is introduced, as shown in Algorithm 1. In this algorithm, we first calculate the single flow similarity, based on the single flow similarity scores between each single flow in the set of flows to be matched and the corresponding single flows in the actual behavioral structure of multiple flows, denoted as $\rho_{flow}(i,I)$. Then, within the matched set of flows M', we calculate the logical similarity scores $l_{flow}(i,j)$ for pairwise combinations of these flows. Using these single flow similarity scores $\rho_{flow}(i,I)$ and logical similarity scores $l_{flow}(i,j)$, we can further calculate the joint similarity scores $\rho_{joint}(i,j)$ between pairs of single flows. Finally, we sum up all the joint similarity scores and normalize them to obtain the final similarity score $\rho(G,G')$ for the multiple flow structure.

In the implementation, for the matched flow set $G'=(V')$ and the given multiple flow structure $G=(V,E)$, this paper introduces three additional similarity metrics to enrich the content of the multiple flow matching degree:

- Size Similarity $\rho_{size}(G,G')$: Describes the similarity in terms of the number of single flows between the set of flows to be matched and the given multiple flow structure. It is calculated as the ratio of the number of single flows contained in both sets, as shown in Eq. 3.

$$\rho_{size}(G,G') = \begin{cases} \frac{size(V)}{size(V')} & \text{if } size(V) < size(V') \\ 1 & \text{if } size(V) = size(V') \\ \frac{size(V')}{size(V)} & \text{if } size(V) > size(V') \end{cases} \quad (3)$$

- Node Similarity $\rho_{point}(G,G')$: Represents the proportion of successfully matched single flows in the set of flows to be matched to the total size of that set. It is calculated as the ratio of the number of successfully matched single flows to the size of the set of flows to be matched, as shown in Eq. 4.

$$\rho_{point}(G,G') = sizeOf(M') \,/\, sizeOf(V') \quad (4)$$

- Main Flow Similarity $\rho_{inevitable}(G,G')$: Describes the matching degree of the set of flows to be matched with the main flow set T in the given multiple flow structure. It is calculated as shown in Eq. 5.

$$\rho_{inevitable}(G,G') = \frac{\sum_{i\in V', t\in T} \rho_{flow}(i,t)}{sizeOf(T)} \quad (5)$$

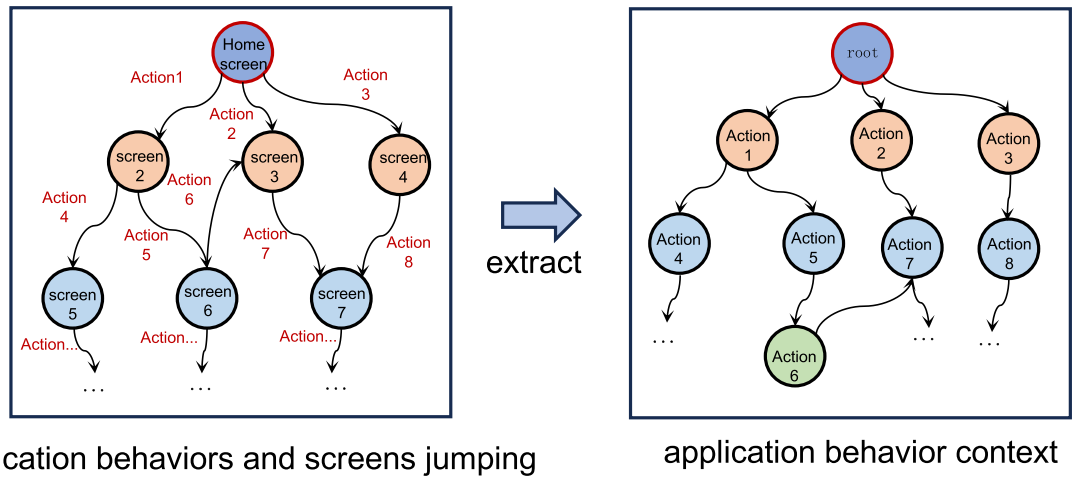


Fig. 4 Extraction of application behavior traffic context

In $\rho_{inevitable}(G, G')$, the concept of the main flow is introduced. The main flow refers to a single flow that appears in every instance of the same behavior's multiple flow structure. The main flow represents a prominent portion of data transmission in that behavior, containing essential characteristics of the application's functional traffic. If the traffic of a behavior includes these consistently appearing main flows, we consider it to have a certain degree of stability. Therefore, this paper introduces an additional main flow similarity $\rho_{inevitable}(G, G')$ to emphasize their impact.

So far, we have calculated four similarity metrics for the set of flows to be matched $G' = (V')$ and the given multiple flow structure $G = (V, E)$, including the multiple flow structure similarity $\rho(G, G')$, size similarity $\rho_{size}(G, G')$, node similarity $\rho_{point}(G, G')$, and main flow similarity $\rho_{inevitable}(G, G')$. These similarities will have varying degrees of impact on the overall matching degree of the multiple flows during the evaluation. Therefore, we define a weight factor for each similarity metric, and compute the final multiple flow matching degree *multiMatchDegree* with Eq. 6:

$$multiMatchDegree = \alpha * (\omega * \rho_{inevitable} + \phi * \rho(G, G')) + \beta * \rho_{point} + \gamma * \rho_{size} \tag{6}$$

Construction of Behavior Characteristics based on Application Context

Application Behavior Context Construction

Mobile applications typically comprise a variety of function screens, including the login screen, main screen, search screen, and results display screen, among others. User interactions within the application prompt transitions between these screens, as illustrated in Fig. 4. The logic governing these screen transitions differs across applications, resulting in unique user behavior contexts within each application. Therefore, by examining the patterns of screen transitions within an application, it is possible to delineate the contextual relationships between different user behaviors. This approach enables the analysis of the contextual connections reflected in the network traffic generated by activities within the application.

In particular, we define the endpoint of a directed edge as the final screen reached after performing corresponding behaviors. This is because some behaviors, while executed, may cause consecutive jumps in the application's screens, which can span multiple screen. The final screen refers to the screen reached after executing the behavior and where no further jumps occur until the completion of that behavior without external interference.

In real-world scenarios, users typically start using an application from its launch activity (Launch Activity) or the "main screen" (the first screen displayed after the application is launched). Therefore, we can set a root node for the directed graph that describes contextual relationships, which is the main screen of the application.

Contextual Relationships Matching

As previously mentioned, contextual relationships fundamentally describe the relationships between multiple streams contained within application behaviors. Therefore, the matching of contextual relationships also relies on the matching of multi-flow relationships. Algorithm 2 describes the process of matching contextual relationships.

Algorithm 2 Context Matching Algorithm

Input: Behavior traffic sequence to be matched Ω, root node $node_{root}$
Output: Matched multi-flow sequence S

$S \leftarrow \emptyset$;
$Node \leftarrow node_{root}$;
while $\Omega \neq \emptyset$ **do**
  $\varpi \leftarrow \Omega.pop()$;
  $m \leftarrow getMatchedEdges(Node, \varpi)$;
  if $m \neq Null$ **then**
    $S \leftarrow S \bigcup m$;
  else
    break;

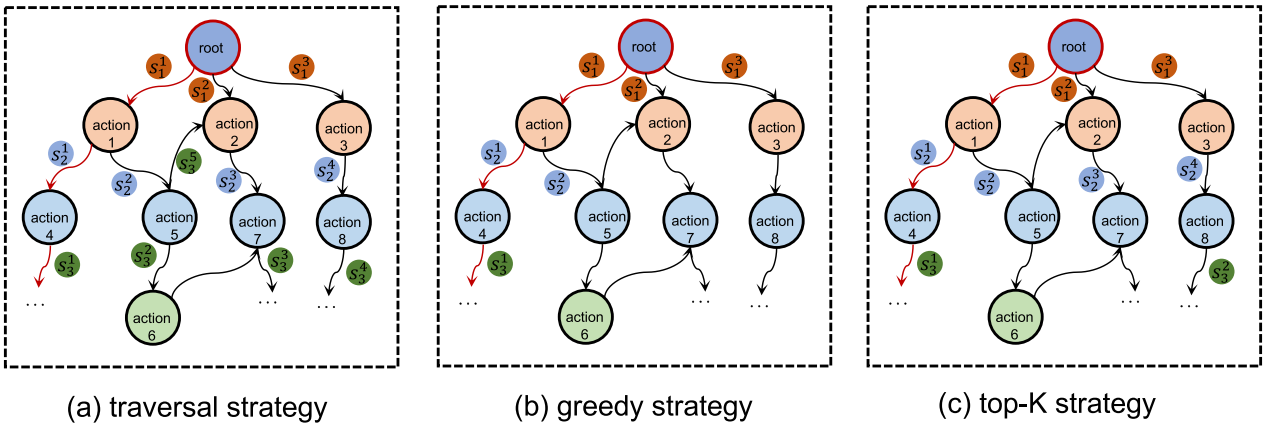


Fig. 5 Strategy of path searching

In essence, contextual relationship matching occurs only after matching a multi-flow relationship. Following the matching of a given multi-flow queue $\Omega = \varpi_1, \varpi_2, \ldots, \varpi_n$, and the root node $node_{root}$ in the contextual relationship graph, for the first element ϖ_1 in the queue, the current node *node* is set to $node_{root}$. Starting from the current node, retrieve all outgoing directed edges (multi-flows), and if the pending multi-flow ϖ_1 matches successfully with one of them, move the current node to the endpoint node pointed to by that directed edge and add i by 1, start the next matching cycle. If the pending multi-flow fails to match with all outgoing directed edges from the current node, the queue of multi-flows fails to match. In particular, for multi-flow queues with higher uncertainty, when a node's match fails, it may not be immediately considered a failed match for the entire multi-flow queue. Instead, the current failed multi-flow can be discarded, and the next multi-flow queue can be popped from the queue for further matching, continuing until the multi-flow queue is empty.

Identification of Encrypted Applications Traffic

This work considers three strategies to identify encryption applications, namely the traversal strategy, the greedy strategy and the Top-K strategy, which are described as follows.

Traversal strategy: For a given traffic to be matched, match all possible paths in the multi-flow context structure in the library, and calculate the path with the highest matching degree as the matching degree of the application corresponding to the multi-flow context structure. The traversal strategy is the most general strategy, as shown in Fig. 5 (a). This strategy calculates the matching degree of the traffic to be matched for all possible paths in each multi-flow context structure, and finally selects the path with the highest matching degree as the current Matching degree of context structure. However, since the multi-flow context structure of most applications is relatively complex, if all valid paths are traversed, a large number of invalid matches will inevitably be generated, resulting in a high time consumption of the traversal strategy.

Greedy strategy: For a given traffic to be matched, only the local best matching result of each layer in the multi-flow context structure in the library is selected as the node of the matching path. The greedy strategy is often used to reduce the time complexity of the algorithm. This strategy tends to select the local optimal solution as part of the global optimal solution. By continuously selecting the local optimal solution, the interference of some low-matching paths is eliminated. thereby achieving faster execution speed. Theoretically, for the traffic to be matched, the greedy strategy will only generate a matching path for the multi-flow context structure, as shown in Fig. 5 (b). Each node on this path is the node with the highest matching degree in the current layer. Therefore, the time consumption of the greedy algorithm is the lowest among existing strategies. However, the greedy strategy has a problem that cannot be ignored, that is, the greedy strategy only looks for an optimal solution in the current layer and ignores the matching of subsequent nodes.

Top-K strategy: For a given traffic to be matched, the Top-K strategy selects the top K multi-flow traffic nodes in similarity from each layer set as subsequent matching paths, as shown in Fig. 5 (c). The Top-K strategy, building upon the concept of local optimality in the greedy strategy, employs the idea of considering locally better solutions for selecting subsequent matching paths. Top-K strategy takes into account the top-K locally optimal nodes to explore the matching of these nodes' subsequent paths. Additionally, the number of next possible

Table 2 The Used Dataset

Applications	Behavior	No. of samples	No. of packets	Size (MB)
VK	Start	20	6,229	2.71
	Upload images	20	8,318	6.07
	Visit photos	20	31,236	19.08
	Send gifts	20	14,806	9.24
	Post media	20	23,277	13.90
Bilibili	Start	20	19,506	19.88
	Visit profile	20	12,407	10.60
	Watch video	20	13021	13.72
	Post moments	20	19,027	21.20
	Fresh homepage	20	11,021	11.26
Facebook	Start	20	7,256	7.30
	Visit profile	20	10,242	8.90
	Upload images	20	12,492	12.35
	Fresh homepage	20	24,175	24.80
Weibo	Start	20	7,752	7.90
	Visit homepage	20	10,923	10.60
	Post media	20	13,989	13.72
	View comments	20	20,924	21.20
	Fresh homepage	20	11,394	11.26
Redbook	Start	20	113,106	120.30
	Visit profile	20	142,027	130.00
	Watch video	20	98,462	108.60
	Post moments	20	20,398	20.32
	Fresh homepage	20	95,019	102.60
Yingyong Bao	Start	20	36,984	38.75
	Application info	20	18,128	18.40
	Download applications	20	39,027	43.26
Total	27 Behaviors	540	841,146	827.92

node decreases as the matching path becomes longer. Therefore, the Top-K strategy continuously reduces the value of K with the increasing path length to accelerate the matching process.

For example, as shown in Fig. 5, starting from the root node. In the traversal strategy, as shown in Fig. 5 (a), the root node traverses all three nodes of the next level, then selects a node from the nodes of the next level in turn. It traverses all the nodes of the next level of the node again until all nodes have been traversed, 12 nodes are traversed in this strategy. The root node traverses three nodes of the next level and then selects the highest node in similarity as the next node. Each layer only selects the node with the highest similarity (including child nodes) as the next node in the greedy strategy; a total of 6 nodes are visited in Fig. 5 (b) with the greedy strategy. In traversal strategy as shown in Fig. 5 (c), top-K nodes in similarity are selected in each layer, where K=3 in the first layer, K=2 in the second layer, and K=1 in the third layer. Finally, a total of 9 nodes are visited.

The evaluation of classification effectiveness

In the evaluation section, we rigorously assess the effectiveness of two key aspects: application behavior traffic and application traffic classification. This evaluation involves a systematic analysis of the performance metrics relevant to each area. For application behavior traffic, we focus on the accuracy and reliability of identifying distinct user activity within the application based on their network traffic patterns. For application traffic classification, we concentrate on the precision, recall, and overall accuracy of correctly categorizing the traffic into predefined classes or types.

Dataset, Evaluation Metrics, and Baselines

For our experimental evaluation, we gathered data on 27 distinct behavioral encrypted traffic across six different application types, resulting in a dataset of 540 behavioral encrypted traffic sample sets (detailed in Table 2). Each behavioral traffic comprises multiple network flows.

Multi-class Confusion Matrix is a tool used in machine learning to evaluate the performance of classification models. It's an extension of the basic confusion matrix (Luque et al. 2019), which is typically used for binary classification tasks. In a multi-class confusion matrix, there are more than two classes, and the matrix helps in visualizing the performance of the algorithm across all these classes. The matrix is a square matrix, where the number of rows and columns equals the number of classes. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. For example, each $cell_{i,j} = \frac{N_{i,j}}{N_i}$, where $N_{i,j}$ is the instances of class i predicted to class j, and N_i is the total instances of class i. Thefore, $\sum_{k=1}^{j} cell_{i,k} = 1$, the sum of cells in each column is 1. The confusion matrix is presented in a tabular format and is essential for evaluating the prediction results of a classification model. It offers a comprehensive measure of the model's performance by illustrating the correlation between actual categories and those predicted by the model.

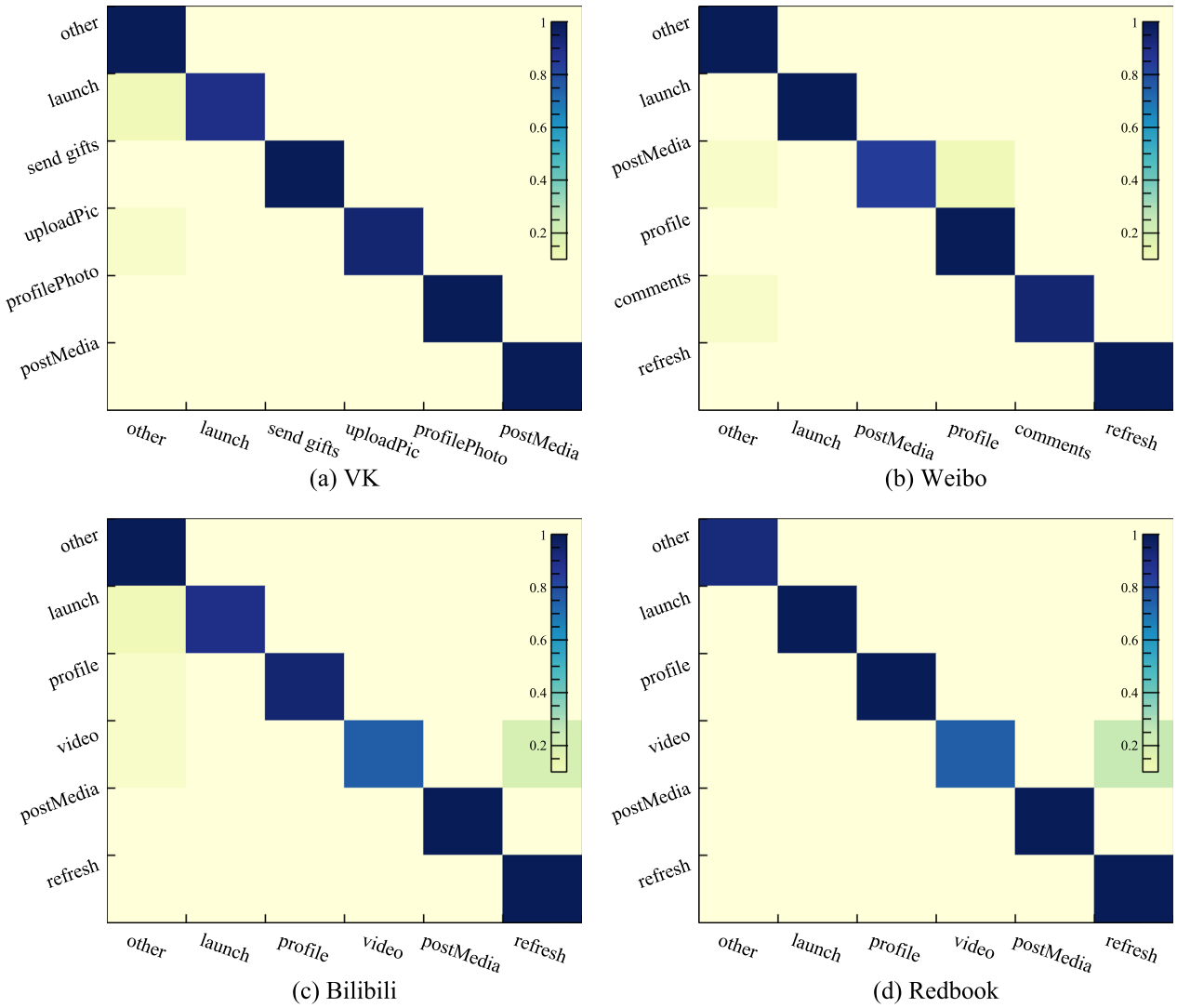


Fig. 6 Confusion matrix of application behavior traffic classification

Accuracy is the proportion of samples that are correctly classified among all samples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

Precision is defined as:

$$Percision = \frac{TP}{TP + FP} \tag{8}$$

and *Recall* is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

The Macro *F1* scores is the average F1-score of all classes. F1 can be defined as follow:

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{10}$$

In the comparative experiment section, we conduct an analysis contrasting the multi-flow traffic classification method with the established single-flow-based classification approach. This comparison aims to ascertain the relative efficacy of the multi-flow-based traffic classification in comparison to the conventional single-flow classification method. The experimental evaluation utilizes the network traffic characteristics delineated in Table 1 as features.

Table 3 Effectiveness on application behavior traffic of each application (Average ACC, Macro PR, RC, and F1)

Applications	Accuracy	Prediction	Recall	F1
VK	0.977	0.985	0.975	0.979
Weibo	0.969	0.974	0.966	0.969
Bilibili	0.938	0.953	0.933	0.938
Redbook	0.946	0.952	0.947	0.944
Average	0.958	0.966	0.955	0.961

In recognition of the often superior classification capabilities of machine learning methods with

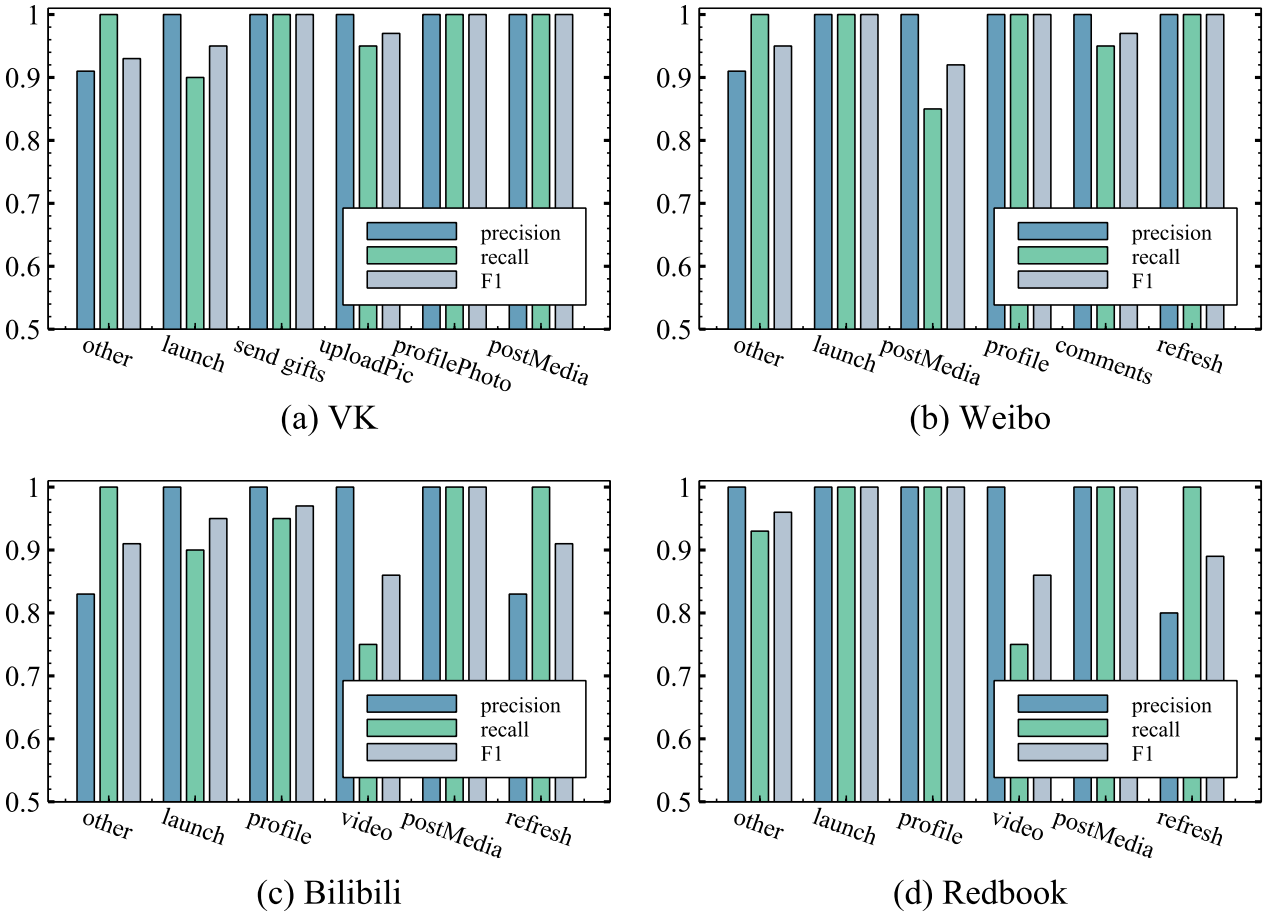


Fig. 7 Results of application behavior traffic classification

low-dimensional features over deep learning approaches, especially in scenarios with limited data complexity (Buchanan 2022; Vogelstein et al. 2021). We opt to utilize machine learning methods as a benchmark in our reference experiment. The selected methodologies for this comparative analysis include:

- Liu et al. (2021), which uses random forest,
- Jadav et al. (2021), which bases logistic regression algorithm,
- Fan et al. (2017), uses Support Vector Machine (SVM) algorithm,
- Afuwape et al. (2021), which is based on K-Nearest Neighbors (KNN),
- Alhawi et al. (2018), which uses Bayesian classifiers,
- Flowprint Van Ede et al. (2020), which uses the communication destion address and TLS certificate as the app's fingerprint, and classifies applications via fingerprint.

These methods are chosen for their effectiveness in handling low-dimensional feature spaces and their proven track record in traffic classification tasks. Through this comparative experiment, we aim to provide a comprehensive understanding of how multi-flow and single-flow traffic classification methods perform under similar conditions and feature sets.

Table 4 Comparison with single flow-based methods and Flowprint on behavior traffic classification within application (Average ACC, Macro PR, RC, and F1)

Algorithems	Accuracy	Prediction	Recall	F1
Liu et al. (2021)	0.716	0.715	0.684	0.699
Jadav et al. (2021)	0.727	0.729	0.680	0.703
Fan et al. (2017)	0.590	0.635	0.523	0.573
Afuwape et al. (2021)	0.640	0.640	0.627	0.633
Alhawi et al. (2018)	0.817	0.804	0.797	0.800
Flowprint Van Ede et al. (2020)	0.452	0.424	0.512	0.463
Multi-flow	0.958	0.966	0.955	0.961

Effectiveness on Application Behavior Traffic Classification

In the context of the dataset composition described earlier, we successfully identified different behaviors within the VK, Weibo, Bilibili, and Redbook applications. The confusion matrices for behavior classification within each application are presented in Fig. 6.

In a multi-class confusion matrix, each cell (i, j) indicates the fraction of samples whose true class is i but are predicted by the model to be in class j. Thus, the main diagonal elements of the matrix signify the percentage of samples accurately predicted by the model, while the off-diagonal elements reflect the model's misclassifications. For instance, 5% of the samples of application *VK* genuinely belonging to the *uploadPic* (representing upload picture behavior) are incorrectly classified as *other* samples in Fig. 6. Conversely, 95% are correctly identified as uploading picture behavior. Within the Weibo application, Around 10% of postMedia traffic is misclassified as profile visiting. The act of posting media with images entails a more complex process. This process not only involves uploading images but also concurrently results in the creation of a new post. Furthermore, accessing the profile screen initiates a dynamic retrieval of images from the album. This overlapping functionality between posting and viewing activities on Weibo can occasionally lead to confusion. The behavioral patterns within applications like Bilibili and Redbook, particularly in activities such as video watching, are prone to ambiguity. This confusion primarily arises from the similarity in their service content structures. In both activities, the behavioral traffic is dominated by streaming media content, encompassing videos, and a mix of static and dynamic images. This resemblance in content delivery and structure makes it challenging to distinctly categorize these behaviors. Figure 7 provides detailed effectiveness metrics of our method for behavior traffic classification across various

Table 5 Comparison with single flow-based methods and Flowprint on application traffic classification (Average ACC, Macro PR, RC, and F1)

Algorithems	Accuracy/Δ	Prediction/Δ	Recall/Δ	F1/Δ
Liu et al. (2021)	0.809/+0.000	0.796/-0.001	0.778/+0.001	0.786/+0.000
Jadav et al. (2021)	0.918/+0.000	0.870/+0.000	0.858/-0.001	0.863/-0.001
Fan et al. (2017)	0.782/+0.000	0.756/+0.000	0.746/+0.000	0.750/+0.000
Afuwape et al. (2021)	0.884/+0.001	0.879/-0.001	0.862/+0.001	0.870/+0.000
Alhawi et al. (2018)	0.902/0.000	0.895/-0.002	0.885/0.001	0.889/-0.001
Flowprint Van Ede et al. (2020)	**0.962**/-0.439	0.953/-0.438	**0.962**/-0.470	0.957/-0.454
Multi-flow	0.955/+0.000	**0.966**/-0.001	0.950/+0.001	**0.957**/+0.000

applications, while Table 3 presents the average performance for each application.

In order to evaluate the effectiveness of the introduced multi-flow approach, this experiment conducts a comparative analysis with single-flow feature-based machine learning methods and multi-flow based method, named Flowprint. The methods selected for this comparison include Random Forest, Logistic Regression, SVM (Support Vector Machine), K-Nearest Neighbor (KNN), and Naive Bayes. These models are developed using the those single-flow features specified in Table 1, applied to our dataset. For Flowprint, we use the same multi-flow features with Flowprint (Van Ede et al. 2020) to represent application behavior traffic and application traffic, and we use the default setting $\tau_{correlation}$=0.1, and $\tau_{similarity}$ =0.9.

The average results of this comparison on behavior traffic classification within application (VK, Weibo, Bilibili, and Redbook), which are detailed in Table 4, reveal that among the chosen single-flow methods, Naive Bayes (Alhawi et al. 2018) emerges as the most accurate, achieving an accuracy of approximately 0.817. However, this accuracy is notably lower than that of the encrypted behavior classification model that utilizes network multi-flow traffic relationships. This disparity underscores the enhanced efficiency of the multi-flow structure in accurately classifying behaviors within encrypted traffic, highlighting its superiority over traditional single-flow feature-based approaches in this specific context. It's noteworthy that the Flowprint (Van Ede et al. 2020) method achieved an accuracy of only 0.452. This outcome can be attributed to the fact that different behaviors within the same application tend to communicate with identical addresses and share the same certificate information. Consequently, in Flowprint, the fingerprints for different behaviors appear identical, leading to the method's inability to accurately distinguish between various behaviors within an application.

Effectiveness on Application Traffic Classification

We utilize a database table to encapsulate the context of application behaviors traffic. This table is specifically designed to store the multi-flow traffic representing the potential subsequent behaviors associated with each application behavior. By organizing behavior multi-flow traffic context in a tabular format, we can effectively capture and query the sequence and interrelation of behaviors within an application. These applications include VK, Weibo, Bilibili, and Redbook. In this experiment, we select 20 traffic sets from the encrypted traffic of each application to serve as positive samples and 30 traffic sets randomly selected from other applications as other samples.

For this study, we evalute our approach and other five distinct machine learning algorithms on the identical encrypted application traffic dataset. The classification results obtained from this testing are systematically presented in Table 5, with bolded values indicating the results of the best-performing method. We deploy the Algrothm 2 with traversal strategy, which yields the most favorable classification outcomes across various metrics, including accuracy, precision, recall, and F1 score. Notably, Flowprint, multi-flow based, achieves the highest accuracy of 0.962, the highest recalll of 0.962. Our method achieves the second-highest accuracy of 0.955 which is lower than the highest accuracy by 0.007, and achieves the highest prediction of 0.966. The accuracy achieved by the our method was 5% higher than that of the best performing single-flow based methods. This margin indicates a substantial improvement and highlights the effectiveness of the multi-stream approach in classifying application traffic, particularly when

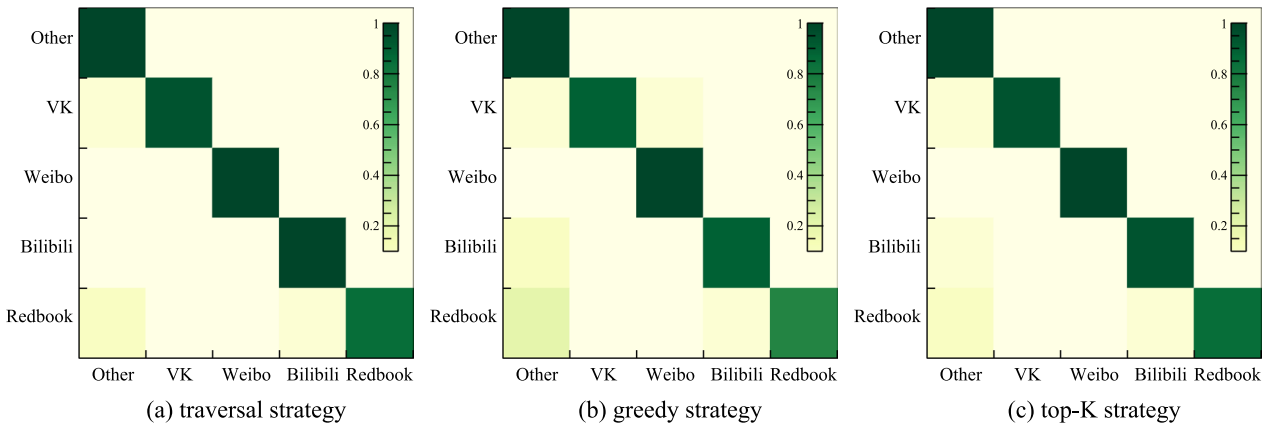


Fig. 8 Confusion matrix of application classification under different search strategies

compared to other machine learning models applied to the same dataset.

To assess the resilience of our model in the face of dynamic changes in network traffic, we devised experiments involving our previously trained models. In these tests, we randomly selected 50% of our test samples for modifications specifically targeting the IP addresses and ports of the traffic samples, as well as the server name domain information contained within the certificates. The purpose of these modifications was to evaluate the model's ability to maintain its performance when confronted with alterations in network traffic features, which are typical in real-world environments. Following these modifications, we utilized the trained models to classify the altered samples. The impact of these feature modifications on the model's performance is detailed quantitatively in Table 5. In this table, the Δ is employed to denote the variation in the model's performance metrics before and after the modifications were applied.

We observed that the classification performance of our method, as well as other single-flow-based classification methods, remained essentially unchanged. This stability is attributed to the fact that these methods rely on flow behavior features and behavior context features, which are less susceptible to changes in network configurations such as IP addresses and certificates. Conversely, the accuracy of Flowprint experienced a significant decline, dropping by 0.439. The underlying reason for this decrease is that Flowprint constructs fingerprints based on the IP address and certificate information. Therefore, whenever there is a change in the service provider's IP address or certificate, the fingerprints generated by Flowprint lose their validity, leading to a marked reduction in its classification accuracy. This observation highlights the vulnerability of methods that depend heavily on static features in dynamic network environments, underscoring the importance of adopting features that are robust to such changes.

The comprehensive results in Table 5 provide a clear demonstration of the superior performance of our proposed method in the context of encrypted application traffic classification.

Comparation under Different Search Strategy

In the realm of network traffic analysis, classification efficiency stands as a crucial metric. Consequently, we employ classification time cost as a key parameter to evaluate the performance of various search strategy in our approach. This metric focuses on quantifying the speed and responsiveness of different search strategy in identifying and classifying network traffic, providing insight into their operational efficiency and practicality in real-world scenarios. By comparing the time taken for classification across different search strategy, we can assess their relative effectiveness and suitability for deployment in network traffic monitoring and management systems.

We utilize the identical sample dataset as employed in Sect. , applying it to the classification task outlined therein. The focus of this evaluation is to ascertain both the classification effectiveness and the time efficiency of our method when implemented under three distinct strategies: traversal, greedy, and Top-K. Figure 8 offers a detailed visual representation of the classification confusion matrices for each of these search strategies. From this figure, we observe notable differences in the effectiveness of each strategy. Under the traversal strategy, the data points are predominantly concentrated along the diagonal of the matrix, indicating a high rate of correct classifications. Conversely, the effectiveness under the greedy strategy is observed to be the least effective. This is evident from the considerable scattering of data points outside the diagonal, indicating a higher incidence of misclassifications. The effectiveness of the TopK strategy, while does not match the effectiveness of the traversal strategy, superior to the greedy strategy.

Table 6 presents a comprehensive overview of the overall classification effectiveness and time consumption associated with our method when implemented using different search strategies. This comparative analysis allows us to assess the balance between efficiency and accuracy for each strategy. It is evident that the traversal strategy outperforms the others in terms of accuracy and F1 score, achieving the highest values of 0.964 and 0.960, respectively. However, this increased accuracy comes at the cost of time efficiency, as the traversal strategy exhibits a 48% higher time consumption compared to the greedy strategy. In contrast, the greedy strategy, while being the most time-efficient, records lower effectiveness with accuracy and F1 values of 0.918 and 0.921. The Top-K strategy offers a near-optimal blend of effectiveness and efficiency. It achieves an accuracy of 0.955 and an F1 score of 0.956, closely rivaling the traversal strategy's performance. Notably, it accomplishes this with 25% less time consumption compared to the traversal strategy. In summary, while the traversal strategy offers the highest classification effectiveness, it is also the most time-consuming. The greedy strategy, on the other hand, is the least effective but the most time-efficient. The TopK strategy provides

Table 6 Evaluation results of different search strategies

	Accuracy	Prediction	Recall	F1	Time cost (s)
traversal strategy	0.964	0.972	0.960	0.960	55
greedy strategy	0.918	0.942	0.910	0.921	37
top-K	0.955	0.966	0.950	0.956	42

a balanced solution, minimizing time consumption without significantly compromising on effectiveness. This comparative analysis underlines the importance of considering both accuracy and efficiency when selecting an appropriate strategy for classification tasks.

Limitation

In discussing the limitations of our approach to network traffic detection, it's important to acknowledge the diverse range of network adversary attacks that exist today. These attacks can involve dynamically changing domain names and server IP addresses, as well as altering the size and timing characteristics of network data packets (Nasr et al. 2021; Li et al. 2023). Through our experiments, we've demonstrated that our method maintains its effectiveness in classifying traffic even after modifications to IP characteristics and certificate information. However, it's critical to note that attacks which modify the size and timing characteristics of network packets pose a significant challenge to our classification accuracy.

These sophisticated attacks can fundamentally alter the behavioral patterns that our model relies on for accurate classification, thereby undermining its effectiveness. For instance, changes in packet size and timing can disrupt the flow behavior features and behavior context features that are essential for our method's ability to correctly identify and classify network traffic. While our model exhibits resilience against changes to static features such as IP addresses and certificates, it is more vulnerable to attacks that manipulate dynamic aspects of network traffic.

This limitation underscores the need for continuous improvement in our method and the development of advanced techniques capable of detecting and adapting to the evolving tactics employed by adversaries. Future research could focus on enhancing the model's robustness against a broader array of attack vectors, including those that involve sophisticated manipulations of packet size and timing. Additionally, exploring the application of graph-based methods in traffic classification could provide valuable insights into complex network interactions and dependencies. By addressing these challenges and integrating advanced analytical frameworks, we can improve the reliability and security of network traffic detection systems in the face of increasingly complex and adaptive network threats.

Conclusion

In this paper, we have embarked on an in-depth exploration of multi-flow application behaviors traffic within the realm of mobile applications. The crux of our work is that we proposed a method that models the multi-flow relationships within applications. We have rigorously evaluated the effectiveness and efficiency of our proposed approach through a series of comprehensive experiments. Our method demonstrated a remarkable accuracy of 0.958 in application behavior traffic classification. Furthermore, it achieved an accuracy of 0.955 in application traffic classification. Notably, these results represent a significant improvement, exceeding the performance of traditional single flow-based network traffic analysis methods by 5.3% in terms of accuracy.

Acknowledgements
This work is supported by the National Key Technologies Research and Development Program of China (grant No. 2022YFC3321102).

Author contributions
GE Mengmeng: Methodology and writing; Feng Ruitao: Validation and writing; Xie Xiaofei: Visualization/ data presentation; Liu Likun: Data Curation; Yu Xiangzhan: Project administration; Sachidananda Vinay: Writing - Original Draft; Liu Yang: Writing - Review & Editing.

Funding
This work is supported by the National Key Technologies Research and Development Program of China (grant No. 2022YFC3321102).

Availability of data and materials
All datasets used be avaliable.

Declarations

Conflict of interest
The authors declare that they have no conflict of interest.

Received: 5 February 2024 Accepted: 26 June 2024
Published online: 20 April 2025

References

Aceto G, Ciuonzo D, Montieri A, Pescapé A (2018) Mobile encrypted traffic classification using deep learning. In *2018 Network traffic measurement and analysis conference (TMA)*, pages 1–8. IEEE

Afuwape AA, Ying X, Anajemba JH, Srivastava G (2021) Performance evaluation of secured network traffic classification using a machine learning approach. Comput Stand Interf 78:103545

Alhawi Omar MK, Baldwin J, Dehghantanha A (2018) Leveraging machine learning techniques for windows ransomware network traffic detection. *Cyber threat intelligence*, pages 93–106,

Bar-Yanai R, Langberg M, Peleg D, Roditty L (2010) Realtime classification for encrypted traffic. In *Experimental Algorithms: 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings 9*, pages 373–385. Springer

Barradas D, Santos N, Rodrigues L, Signorello S, Ramos Fernando MV, Madeira A (2021) Flowlens: Enabling efficient flow classification for ml-based network security applications. In *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium*

Bartos K, Sofka M, Franc V (2016) Optimized invariant representation of network traffic for detecting unseen malware variants. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 807–822, Austin, TX, August. USENIX Association

Borders K, Springer J, Burnside M (2012) Chimera: A declarative language for streaming network traffic analysis. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 365–379, Bellevue, WA, August. USENIX Association

Buchanan S (2022) *Deep networks through the lens of low-dimensional structure: towards mathematical and computational principles for nonlinear Data*. Columbia University

Bujlow T, Carela-Español V, Barlet-Ros P (2015) Independent comparison of popular dpi tools for traffic classification. Comput Netw 76:75–89

Conti M, Mancini Luigi V, Spolaor R, Verde Nino V (2016) Analyzing android encrypted network traffic to identify user actions. IEEE Trans Inform Forensics Sec 11(1):114–125

Dong S, Xia Y, Peng T (2021) Network abnormal traffic detection model based on semi-supervised deep reinforcement learning. IEEE Trans Netw Serv Manage 18(4):4197–4212

Draper-Gil Gerard, Lashkari Arash Habibi, Mamun Mohammad Saiful Islam, Ghorbani Ali A (2016) Characterization of encrypted and vpn traffic using time-related. In *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, pages 407–414. sn

Fan Z, Liu R (2017) Investigation of machine learning based network traffic classification. In *2017 International Symposium on Wireless Communication Systems (ISWCS)*, pages 1–6

Feng X, Ling X, Zheng H, Chen Z, Yiwen X (2019) Adaptive multi-kernel svm with spatial-temporal correlation for short-term traffic flow prediction. IEEE Trans Intell Trans Syst 20(6):2001–2013

Holland J, Schmitt P, Feamster N, Mittal P (2021) New directions in automated traffic analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 3366-3383, New York, NY. Association for Computing Machinery

Jacob G, Hund R, Kruegel C, Holz T (2011)JACKSTRAWS: Picking command and control connections from bot traffic. In *20th USENIX Security Symposium (USENIX Security 11)*, San Francisco, CA, August. USENIX Association

Jadav N, Dutta N, Sarma Hiren KD, Pricop E, Tanwar S (2021) A machine learning approach to classify network traffic. In *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–6

Li W, Zhang X-Y, Bao H, Yang B, Li Z, Shi H, Wang Q (2023) Prism: Real-time privacy protection against temporal network traffic analyzers. IEEE Trans Inform Forensics Sec 18:2524–2537

Li W, Zhou S, Li R, Zhang K, Qichao X (2020) Abnormal crowd traffic detection for crowdsourced indoor positioning in heterogeneous communications networks. IEEE Trans Netw Sci Eng 7(4):2494–2505

Li Y, Feng R, Chen S, Guo Q, Fan L, Li X (2021)Iconchecker: anomaly detection of icon-behaviors for android apps. In *2021 28th Asia-Pacific Software Engineering Conference (APSEC)*, pages 202–212

Li H, Hu H, Gu G, Ahn G-J, Zhang F (2018) vnids: Towards elastic security with safe and efficient virtualization of network intrusion detection systems. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–34

Lin X, Xiong G, Gou G, Li Z, Shi J, Jing Y (2022) Et-bert: a contextualized datagram representation with pre-training transformers for encrypted traffic classification. Proc ACM Web Conf 2022:633–642

Liu L, Wang P, Lin J, Liu L (2021) Intrusion detection of imbalanced network traffic based on machine learning and deep learning. IEEE Access 9:7550–7563

Liu C, He L, Xiong G, Cao Z, Li Z (2019) Fs-net: A flow sequence network for encrypted traffic classification. In *IEEE INFOCOM 2019-IEEE Conference On Computer Communications*, pages 1171–1179. IEEE

Luque A, Carrasco A, Martín A, de Las HA (2019) The impact of class imbalance in classification performance metrics based on the binary confusion matrix. Pattern Recognit 91:216–231

Mirsky Y, Doitshman T, Elovici Y, Shabtai A (2018) Kitsune: An ensemble of autoencoders for online network intrusion detection. In *The Network and Distributed System Security Symposium (NDSS) 2018*

Nakibly G, Schcolnik J, Rubin Y (2016) Website-targeted false content injection by network operators. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 227–244, Austin, TX, August. USENIX Association

Nasr M, Bahramali A, Houmansadr A (2021) Defeating {DNN-Based} traffic analysis systems in {Real-Time} with blind adversarial perturbations. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2705–2722,

Shafiq M, Yu X, Laghari A A, Yao L, Karn N K, Abdessamia F (2016) Network traffic classification techniques and comparative analysis using machine learning algorithms. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pages 2451–2455

Shang Q, Tan D, Gao S, Feng L (2019) A hybrid method for traffic incident duration prediction using BOA-optimized random forest combined with neighborhood components analysis. J Adv Trans 2019:1–11. https://doi.org/10.1155/2019/4202735

Shapira T, Shavitt Y (2021) Flowpic: a generic representation for encrypted traffic classification and applications identification. IEEE Trans Netw Serv Manage 18(2):1218–1232

Shen M, Liu Y, Zhu L, Xiaojiang D, Jiankun H (2021) Fine-grained webpage fingerprinting using only packet length information of encrypted traffic. IEEE Trans Inform Forensics Sec 16:2046–2059

Shen M, Zhang J, Zhu L, Ke X, Xiaojiang D (2021) Accurate decentralized application identification via encrypted traffic analysis using graph neural networks. IEEE Trans Inform Forensics Sec 16:2367–2380

Sirinam P, Imani M, Juarez M, Wright M (2018) Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1928-1943, New York, NY, USA. Association for Computing Machinery

Smart home trends to watch in 2022 and beyond (2022)

Sun Y, Edmundson A, Vanbever L, Li O, Rexford J, Chiang M, Mittal P (2015) RAPTOR: Routing attacks on privacy in tor. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 271–286, Washington, D.C., August. USENIX Association

Taylor Vincent F, Spolaor R, Conti M, Martinovic I (2016)Appscanner: automatic fingerprinting of smartphone apps from encrypted network traffic. In *2016 IEEE European Symposium on Security and Privacy (EuroS &P)*, pages 439–454

Van Ede T, Bortolameotti R, Continella A, Ren J, Dubois Daniel J, Lindorfer M, Choffnes D, Van Steen M, Peter A (2020) Flowprint: semi-supervised mobile-app fingerprinting on encrypted network traffic. In *Network and distributed system security symposium (NDSS)*, volume 27,

Vogelstein JT, Bridgeford EW, Tang M, Zheng D, Douville C, Burns R, Maggioni M (2021) Supervised dimensionality reduction for big data. Nature Commun 12(1):2872

Wang P, Chen X, Ye F, Sun Z (2019) A survey of techniques for mobile service encrypted traffic classification using deep learning. IEEE Access 7:54024–54033

Wang W, Zhu M, Wang J, Zeng X, Yang Z (2017) End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *2017 IEEE international conference on intelligence and security informatics (ISI)*, pages 43–48. IEEE

Wang W, Zhu M, Zeng X, Ye X, Sheng Y (2017) Malware traffic classification using convolutional neural network for representation learning. In *2017 International conference on information networking (ICOIN)*, pages 712–717. IEEE

Wang T, Goldberg I (2017) Walkie-talkie: an efficient defense against passive website fingerprinting attacks. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1375–1390, Vancouver, BC, August. USENIX Association

Wang J, Cao Z, Kang C, Xiong G (2019) User behavior classification in encrypted cloud camera traffic. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE

Xie M, Jiankun H, Han S, Chen H-H (2012) Scalable hypergrid K-NN-based online anomaly detection in wireless sensor networks. IEEE Trans Parallel Distrib Syst 24(8):1661–1670

Zheng W, Gou C, Yan L, Mo S (2020) Learning to classify: a flow-based relation network for encrypted traffic classification. Proc Web Conf 2020:13–22

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.